DEPARTMENT OF ENGINEERING MATHEMATICS

Vector-Quantised Variational Auto-Encoders for Multimodal Sensor Fusion

In the context of Human Behaviour Detection

Kiran Sankar Edassery Jayanthan

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering.

Thursday 28th August, 2025

Supervisor: Dr. Robert Piechocki

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of MSc in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Kiran Sankar Edassery Jayanthan, Thursday 28th August, 2025

Contents

List of Figures

Ethics Statement

This project did not require ethical review, as determined by my supervisor, Dr. Robert Piechocki

Abstract

OPERAnet is a multimodal activity recognition dataset acquired from radio frequency and vision-based sensors. It is intended to be used to advance Human Activity Recognition (HAR) using any of the provided sensor information. In this project, the focus will be on the Passive Wi-Fi radar (PWR) built upon a Software Defined Radio (SDR) Platform. This part of the dataset provides multiple Doppler spectrograms with 4 antennas corresponding to 1 reference and 3 surveillance channels. The PWR dataset provides two parameters (relative range and Doppler frequency shift) by correlating the signal from two channels each (surveillance and reference).

With the given dataset, it is possible to employ a variety of methods to utilise machine learning for HAR from this dataset. Most of the reviewed literature is on ways of dealing with the multimodality of the data. This study focuses on replicating some of the easier methods initially and then gradually introducing various methods to increase the complexity of the model (by introducing multiple channels and multimodality of information, for example) to derive insights that would be useful in later stages of the problem. To that effect, the literature deals with multimodal sensor fusion in the decision, feature and latent representation space.

Labelling time-series data (In this case, the data is not strictly time-series, but derived from time-series data via mathematical modelling) is a time-consuming and expensive process. The OPERANet does provide sufficiently labelled data, but the replication of the quality and quantity of such well-labelled data is not practical. Most data in the real world would also be much noisier than data collected in a controlled environment. As such, supervised learning methods remain largely impractical for real-world applications involving such data.

This study's goals are two-fold. First, it will focus on the re-implementation of these older (usually supervised learning) methods for comparison of relative ease-of-implementation, computational complexity and result verification. This will allow for the creation of data pipelines that are consistent across implementations, and their contributions to the results can then be disregarded in the latter stages of the study. This will also give a better idea as to the amount of preprocessing and pipeline-specific changes that would have to be made when implementing the more complex algorithms, which are usually disregarded in studies as preprocessing steps. Secondly, the study will focus on investigating the benefits, if any, of using Vector Quantised Variational Auto-Encoders in the context of such data in matters of classification.

The supervised learning methods were evaluated via a basic implementation of a ResNet18 pipeline. This was done by creating images through chunks of spectrogram data with the 3 different channels encoded into the Red, Green and Blue channels to make images. These images were then fed to the system with their respective labels. The results are extremely promising, as expected of an architecture that is very good at the classification of image data, even abstract ones like spectrograms. However, their effectiveness and viability are questioned and broken down for context. Next, two implementations of a Vector-Quantised Variational Auto-Encoder (VQ-VAE) were created to then evaluate the model via the inherent clustering in the latent space. The results are then used to break down how viable the latent space clustering is, even with untuned hyperparameters. Comparisons are drawn between quantised and non-quantised outputs, single-channel and multi-channel outputs and multiple codebook dimensionalities as well. It was concluded that multi-modal VQ-VAEs vastly outperform single channel and single modal implementations even while training for half the epoch cycles and show considerable promise in the field.

Supporting Technologies

This section lists the libraries, technologies and software that have been used throughout the study.

- Version control and code (and result) backups were done using GitHub.
- Most of the programming, data manipulation and processing were done using Python.
- The libraries that were primarily used for the aforementioned processing steps are *pandas*, *numPy*, *scikit-learn*, *tqdm*, *collections*, *pickle*, *random*, and *umap* which are all open-source libraries.
- The GPU-assisted machine learning protocols were accomplished via *PyTorch*(the NVIDIA CUDA versions) and its associated libraries.
- Results were plotted via the use of *seaborn* and *matplotlib*
- I used Google Colab to run most of the performance-intensive GPU-heavy workloads. The free T4 GPU runtime environment was used for most of the implementations.
- I used LaTeX to format my thesis, via the online service *Overleaf*.

Notation and Acronyms

Standard and non-standard acronyms used throughout this document are detailed here for convenience.

AoA	:	Angle of Arrival
AWS	:	Amazon Web Services
BoW	:	Bag of Words
CNN	:	Convolutional Neural Network
CPU	:	Central Processing Unit
CSI	:	Channel State Information
CV	:	Cross-Validation
EC2	:	Elastic Compute Cloud
ELBO	:	Evidence Lower Bound
GPU	:	Graphics Processing Unit
HAR	:	Human Activity Recognition
KLD	:	Kullback-Leibler Divergence
KNN	:	K-Nearest Neighbours
MLP	:	Multi-Layer Perceptron
MSE	:	Mean Squared Error
NIC	:	Network Interface Controller
PCA	:	Principal Component Analysis
PWR	:	Passive Wi-Fi radar
RADAR	:	RAdio Detection And Ranging
RAM	:	Random Access Memory
RF	:	Radio Frequency
RGB	:	Red-Green-Blue
RNN	:	Recurrent Neural Network
S3	:	Simple Storage Service
SDR	:	Software-Defined Radio
SNR	:	Signal-to-Noise Ratio
STE	:	Straight Through Estimator
STFT	:	Short Time Fourier Transform
SVM	:	Support Vector Machine
TCN	:	Temporal Convolutional Network
TDoA	:	Time Difference of Arrival
ToF	:	Time of Flight
t-SNE	:	t-Distributed Stochastic Neighbour Embedding
UMAP	:	Uniform Manifold Approximation and Projection
UWB	:	Ultra-Wideband
VAE	:	Variational Autoencoder
VQ-VAE	:	Vector-Quantised Variational Autoencoder
VRAM	:	Video Random Access Memory

Acknowledgements

I would like to thank my supervisor, Dr Robert Piechocki, for his guidance. He helped me with almost all the phases of research and implementation and gave me a lot of insights into the research-based nature of this project and how to tackle problems that arose within it. I would also like to thank Dr Junaid Bocus for helping me with references and material, and helping me with the OPERANet dataset. His help in sharing the codebase for the multimodal VAE was incredibly important for my understanding of the problem statement and dataset in a programmatic manner.

I am deeply grateful to my friends, family, and colleagues whose support and encouragement kept me on track and made this project possible. Their help was invaluable to the quality of this work.

And finally, I would like to thank myself for not giving up on me, trying so hard through so many problems and through so many other things that could have impeded this study's progress and coming out with something that I would be proud of in the future.

Chapter 1

Introduction

This chapter is intended to give an idea of the project context, its motivations, and challenges. A high-level overview of the intent of the project is also included here. A concise bullet point list to summarise the aims, objectives and achievements of the work is included at the end of this chapter for the reader's convenience.

1.1 Background and Context

Human Activity Recognition (HAR) is, as the name implies, a field that is concerned with the detection and classification of human activity via sensor data. This is usually done in everyday life via smart wearables (like smart watches), AI-assisted cameras in houses, and other compound sensors like infrared sensors and RGB cameras in Microsoft Kinect. It is used for a variety of reasons, from the mundane like video games and context-aware interfaces to more important ones like assistive monitoring, and human-aware robotics[5]. As such, this can be both a field with high commercial as well as societal impact.

Currently, HAR is done via the use of sensors that are more or less specific to their use-case, such as smart watches for exercise tracking, sleep tracking and so on and devices like Microsoft Kinect for gesture tracking (which is more narrowly scoped than the broader field of HAR). However, the issues with such designs are apparent in the fact that, in an assistive monitoring situation, the user can not be expected to always be aware enough to wear and use such devices for their intended purpose.

This is where opportunistic radar succeeds where wearable devices do not. Opportunistic radar refers to the usage of existing radio signals to detect and track targets (people) via their reflections. It follows that Opportunistic Wi-Fi radar is a system that leverages existing Wi-Fi signal clouds to detect and track targets. Wi-Fi is an extremely prevalent technology in most modern homes and cities and grants us an ever-present radio "cloud" with which to detect and track human activity. It is also device-free, non-contact, leverages existing infrastructure, and is robust to most environmental conditions like lighting and ambient noise. It has been shown to produce viable gesture detection using minimal devices and no human instrumentation [16].

To use this passive Wi-Fi radar is not a simple thing, however, as it poses a lot of computational considerations. For one, such behaviour is usually studied by changes in Radio Frequency (RF) induced by changes in the objects in the environment. Here, the "interfering objects" are assumed to be humans, and then, depending on the changes induced in RF signals, we try and estimate the rate of change of motion (Doppler/velocity) and the range of motion that causes such variation in different frequency channels and use that to infer the behaviour of the person within the sensor setup. To that effect, spectrograms are an easy way to visualise such changes. This is because it is a time-frequency representation and such representations, by inherent nature, make it easy to capture these discriminative motion signatures in a plottable manner [6].

The issue with the usage of such spectrograms, however, is due to the inherent nature of the signals themselves, causing them to be noisy around various data points and the difficulty with which to extract features from such high-dimensional, noisy data. There have been multiple implementations that get around this problem using machine learning algorithms. Supervised machine learning algorithms, like ResNETs, can be used to learn patterns in the image representations of the spectrograms and can then

be reliably used to label later data. However, these methods come with their drawbacks. Supervised learning to that degree would require a lot of labelled data that is extremely hard to acquire for every specific scenario (ambient noise, environmental objects, etc.). It would also be incredibly hard to find, label and then train models on such data on a scale that is viable enough to be applied in such scenarios.

To circumvent these limitations, unsupervised learning methods can be used to naturally learn latent-space representations that are clustered and then used for downstream classification. Due to the high dimensionality of the feature space and the need to represent the data in a latent space that is of lower dimensionality as well as learnable to be clustered, Vector Quantised Variational Autoencoders (VQ-VAEs) are postulated to be a viable alternative to conventional supervised learning methods. It can be used to either learn a latent space that can then be used for classification without the overhead of acquiring fully labelled data, as in the case of traditional models [21].

1.2 Problem Statement

Currently existing HAR pipelines face trade-offs as explained in section 1.1. Cameras suffer from privacy, consent and sightline limitations, wearables suffer from compliance, placement, and wearer comfort issues, and conventional radar requires dedicated hardware and costs time, money and effort to deploy. Wi-Fi-based HAR reduces cost and intrusiveness but struggles with domain shifts, low Signal-to-Noise ratio (SNR) and extremely limited labelled data. Spectral features can be noisy and redundant, making generalisation challenging. There is a requirement for data-efficient representations that stabilise across multiple subjects and environments and still remain discriminatory for relevant activity classes with weakly labelled data.

1.3 Research Aim and Objectives

To tackle this problem, this project aims to use VQ-VAEs to create data-efficient latent spaces that are focused on both generalizability as well as classifiability under weak supervision. The primary objective of this project is to create and analyse a VQ-VAE implementation for HAR using the OPERANet dataset [3].

The Wi-Fi spectrogram data given in [3] is specifically from the Passive Wi-Fi radar (PWR) part of the study and requires a fair amount of cleaning and preprocessing. Especially the contiguous block separation (segmenting continuous recordings into activity blocks), normalisation and readability of the data (since most of it is `.mat` files and would require some data wrangling to get meaningful data out to be processed by libraries in Python).

This is followed by the design and implementation of a simple supervised learning method (ResNet) to create a benchmark. For this, the three-channel spectrogram data were transformed into RGB pictures that were then fed into the network. Once the benchmark was created, the final VQ-VAE implementation was used to learn latent representations. This was then followed by the final part of the project, which was to use this trained latent space to accomplish HAR via light classifiers on the latent space itself.

1.4 Contributions

The VQ-VAE implementation proposed and implemented in this project will include a data pipeline that can reliably turn spectrograms into discrete codebooks, improving data efficiency and downstream classifier simplicity. This is because a trained VQ-VAE model's encoder output would be of a much smaller dimensionality than the full input and thus easier and faster to classify if there is enough information about the clustering behaviour of the latent space.

There are some training methods used throughout the project to increase VQ-VAE performance that were previously proposed. Usage of such methods has yielded results that are of interest to such studies involving VQ-VAE implementations and training [12]. Some observations correlate metrics like perplexity and code usage with class separability.

The end goal of the project benchmarks the performance versus standard spectral features and direct CNN baselines. This study would grant us some insights into the benefits and the applicability of each of these methods in this specific domain of HAR.

1.5 Report Structure

The remainder of the report is as follows - Chapter 2 expands on the background, the current literature and bodies of work. Chapter 3 details the data (origins, content, exploration) and the methods used for data manipulation, preprocessing and the entirety of the methodology. Chapter 4 presents results and the analysis of said results. It covers limitations of the results, the reasons for their current state, and their implications. Chapter 5 is the conclusion and outlines future avenues of work that can be done on the topic of this project.

1.6 Introduction Summary

- **Aims**: Data-efficient HAR with VQ-VAE
- **Objectives**: Data wrangling - ResNet baseline - VQ-VAE training - latent-space classification
- **Achievements**: Codebook learning with robust usage and perplexity, benchmarks vs spectral/CNN baselines

Chapter 2

Background

This chapter is intended to compile and summarise a lot of topics into one to give a more technical understanding of the background and underlying context. As such, it will include some rephrasing of things already discussed in Chapter 1. This chapter is intended to give a more academic overview of the current technologies, their strengths, shortcomings and the technical reasoning why the proposed solutions within this project make sense in an objective manner.

2.1 Human Activity Recognition Overview

As explained in section 1.1, currently, the most popular hardware used for HAR suffers from more than one issue. Vision (cameras) is the most useful sensor type for such a use case. It boasts a very high dimensionality-information ratio. However, privacy concerns and a sensitivity to lighting conditions bog down its applicability in a vast number of situations. Wearables are accurate but require a lot of compliance checks and are generally expensive if there are multiple parties of interest, since each individual would have to be instrumented as a single target [19]. Using Wi-Fi for HAR, however, is highly sensitive to multipath and environmental variation. Moreover, the system observes only 2D RF signatures (range–Doppler/CSI spectrograms), which are low-dimensional projections of the underlying 3D motion. Inferring actions from these measurements is an ill-posed inverse problem.

The opportunistic Wi-Fi radar concept, however, reuses commodities that are already widespread and used in almost all scenarios, ranging from individual houses to workplaces and busy streets. This allows for non-intrusive HAR in homes and clinics, which is where concerns of privacy and compliance are higher. The concept of this has to be explained a bit further, including the methods by which these systems can be implemented. This discussion of Wi-Fi sensing would also benefit from discussing another way of Wi-Fi sensing - Channel State Information (CSI). Here, CSI is built upon the principles of Wi-Fi, while PWR is built upon the principles of radar.

CSI grants access to information concerning amplitude and phase information that can then be used to infer the Angle of Arrival (AoA), Time of Flight (ToF) and Time Difference of Arrival (TDoA). This is then used to create a feature space that can then be used for analysis and HAR in this specific case. In contrast, the PWR system, leveraging radar principles, relies on the estimation of relative velocity between the object and antenna using signals reflected off a surveillance area. In a previous study, the performance of PWR was found to be better than that of CSI in the same monostatic layout [14]. And since most Wi-Fi access points are static after installation, it stands to reason that the usage of PWR should be encouraged.

2.2 Opportunistic Wi-Fi radar and Spectrograms

The typical signal chain for a PWR system is given in Figure 2.1. The raw Radio Frequency (RF) signals are denoised, calibrated as per the environment and ambient requirements. The Time to Frequency shift is usually done via Short Time Fourier Transform (STFT). This would give us the spectrogram data (PWR data in OPERANet). This can then be normalised or used as is for analysis and further exploration.

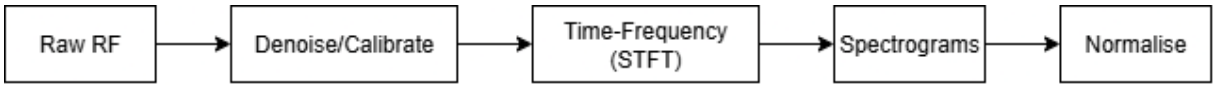


Figure 2.1: PWR Architecture

In the case of the dataset used in this specific project, this basic pipeline holds true, but more technical details are paraphrased here for the convenience of the reader. The CSI data is captured from Intel 5300 NICs at a high packet rate. The PWR system uses a reference plus surveillance channel configuration to form Doppler spectrograms at a lower measurement rate. All streams are time-aligned at a rate of ¡20ms. Direct signal interference from the single reference channel on the 3 surveillance channels was eliminated via the implementation of the CLEAN algorithm [9].

It has been observed that human motion creates characteristic micro-Doppler patterns that are compact and discriminative in this feature space, supporting classifiers and other machine learning approaches [3].

2.3 Deep Learning for HAR

Currently, the norm is to use CNNs of various kinds in the case of spectrogram-like inputs (vision or RF). There are multiple studies showcasing the effectiveness of such methods in capturing spatio-temporal patterns [20]. Recurrent Neural Network (RNN) and Temporal Convolutional Network (TCN) layers are effective in dealing with signals with temporal context. This has been the standard for action segmentation and sequence modelling due to their long effective memory and parallelism [2] [13].

There have also been a lot of advancements in using transformers to model long-range dependencies without convolutions. This convolution-free architecture would theoretically allow these networks to "see" the entire spectrogram at once and model their long-range dependencies without falling into the local receptive fields problem faced by normal CNNs. This is especially useful for spectrograms, which are not true images and might be better modelled via global context.

In the current context of RF/Wi-Fi HAR, deep models on CSI/PWR spectrograms mirror these trends but might still face cross-room issues due to the generalisation issues mentioned in section 1.2. Also, the creation of labelled data for such systems comes with a lot of time and money expenditure. This dissuades the usage of fully supervised systems as the creation itself is expensive and time-consuming, and its applicability to multiple configurations and scenarios is questionable. This does not mean that they would not show high in-domain accuracy.

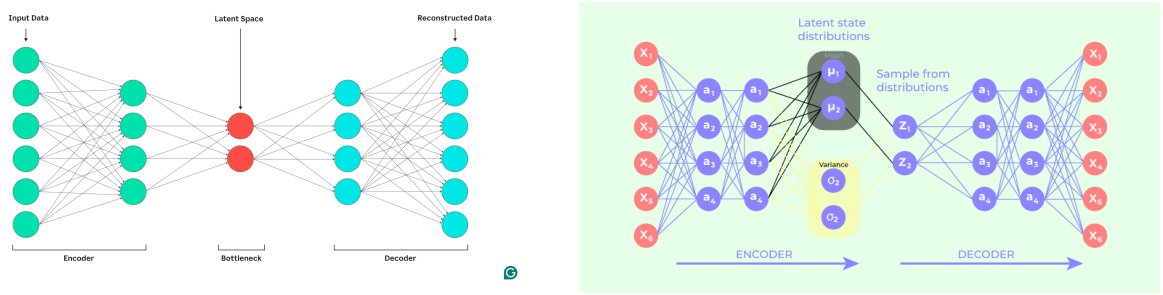
A key pain point in opportunistic passive Wi-Fi radar HAR is label scarcity under multiple environments, layouts, views, and bandwidths/packet rates. This has motivated the usage of generative or self-supervised approaches that can first learn a compact latent space from unlabelled data and then use it for downstream classification tasks. A representative example is latent-space fusion to build a multimodal generative model and using the learnt latent space for classification. This was shown to be advantageous in the case of missing or noisy modalities [15].

2.4 VQ-VAEs: Theory and Relevance

This section is intended to give a theoretical understanding of VQ-VAEs. This has been deemed to be important because of the relative novelty of the technology. To start with VQ-VAEs, it is important to understand the principles of autoencoders and VAEs first.

2.4.1 Autoencoders

Autoencoders (Figure 2.2a) are based on the principle that an encoder-decoder pair with a bottlenecked latent space will learn weights in such a way as to learn to represent the input in the latent space as efficiently as possible if loss is calculated as the difference between the input of the encoder and the output of the decoder. That is, the encoder learns to fill the latent space in such a way as to enable efficient reconstruction. This is facilitated by minimising the reconstruction loss, typically mean-squared error. Mathematically, consider the encoder to be $z = f_\phi(x)$ and decoder to be $\hat{x} = g_\theta(z)$. The objective would be to minimise the Mean Squared Error (MSE), $||x - g_\theta(f_\phi(x))||^2$.



(a) Autoencoder Architecture

(b) Variational Autoencoder Architecture

Figure 2.2: Encoder Architecture. Images sourced from [7] [18]

There are multiple variants to this, such as denoising autoencoders, which specialise in the reconstruction of noisy inputs, contractive autoencoders that encourage local invariance and thus are robust to small variations in the training set, or sparse autoencoders, which constrain average activations and are efficient at learning latent spaces. For all their effectiveness, however, autoencoders are deterministic in that a specific input will always flow through the encoder-decoder architecture to give a specific output. The latent space is almost always unstructured due to the relative unpredictability of how gradients flow and weights are updated. Due to the deterministic way in which they compute outputs, autoencoders can also overfit to near-perfect identity without some controlled training. This also means that for this project, they are not useful as they come with the same labelled data hunger that most CNNs do, but with a much higher overfitting probability.

2.4.2 Variational Autoencoders

To remove this issue and to make the setup generative in nature, Variational Autoencoders were introduced [11]. These, even though their architectural similarity to normal encoders (Figure 2.2b), are probabilistic generative models. They pair a decoder $p_\theta(x|z)$ with a simple latent prior $p(z)$ (typically $\mathcal{N}(0, I)$, aka a normal distribution) and learn an encoder $q_\phi(z|x)$ to approximate the posterior. This is a modification of the basic autoencoder architecture with a stochastic phase in between. This introduces the problem of propagating gradients across a stochastic space, which is not possible. So there is the introduction of the reparameterization trick to make stochastic expectation differentiable. That is,

$$z = \mu_\phi(x) + \sigma_\phi(x) \odot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I) \tag{2.1}$$

This is done to make the gradients of expectation into an expectation of gradients, which does allow for backpropagation. Training is supposed to maximise the ELBO on the log-likelihood:

$$\mathcal{L}(\theta, \phi; x) = \underbrace{\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)]}_{\text{reconstruction term}} - \underbrace{\mathrm{KL}(q_\phi(z \mid x) \,\|\, p(z))}_{\text{regularizer}}. \tag{2.2}$$

With Gaussian decoders, the reconstruction term would correspond to MSE, while binary cross-entropy is used for other discrete versions like the Bernoulli decoder. The KL term for the former has a closed form, which gives a stable training and a principled trade-off between latent compression and fidelity of output.

The full derivation of the loss function and its dependencies is too long to be included in this thesis. The derivation hinges on the application of the Kullback-Leibler Divergence (KL Divergence), variational Bayes inference and Evidence Lower Bound (ELBO) calculation due to the KL divergence being greater than or equal to 0 [11]. Some of these concepts are used in the derivation of the VQ-VAE loss function, and that is explained more in the next section.

2.4.3 Vector-Quantized Variational Autoencoders

VQ-VAEs are the final link in this long exposition covering encoder-decoder pairings. It is also the primary research topic in this thesis, and as such, there will be a deeper dive into the mathematics behind VQ-VAEs given here. The basic architecture of the VQ-VAE (Figure 2.3) is similar to the "encoder-decoder pairing with latent space in between" principle of VAEs and basic autoencoders. Where it differs, however,

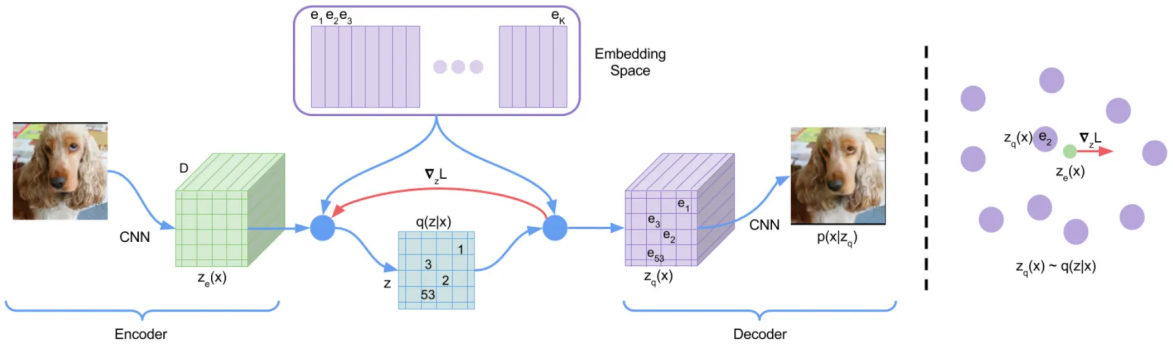


Figure 2.3: VQ-VAE architecture. Sourced from [21]

is the discrete codebook, or the quantisation phase in the latent space part. In a basic encoder, this part is just a simple bottlenecked latent node space. In a VAE, it is a stochastic sampler. However, in the case of the VQ-VAE, it is a quantizer that maps encoder output into discrete codes from a pre-defined codebook.

In other words, the encoder reads an input and outputs a D dimensional $z_e(x)$ that is then mapped to one of the codes $(e_1, e_2, ..., e_K)$ from the codebook creating a quantized output $z_q(x)$ which then goes into the decoder and then gives output $\hat{x}$ which is then used to calculate the reconstruction loss. The code mapping is based on whichever code is "closest" to the encoder output. This gives us a discrete latent space in principle and in practice, which is more manageable and avoids the posterior collapse found in VAEs.

In VAEs, the prior, ie, $p_\theta(z) = \mathcal{N}(\mu, \sigma)$. In a VQVAE, it is quantised and deterministic because it's always the closest neighbour to an output. So, $p(z_q = e_k) = 1/K$, where K is the number of codebook vectors available. This is the probability of the quantised output z_q being mapped to a specific e_k. It is important, however, to remember that this does not mean there is an equal chance of an output being mapped to any k. In fact, the quantisation probability is

$$\begin{aligned} q(z_q = e_k|x) &= 1, \quad \text{for the minimum distance k} \\ q(z_q = e_k|x) &= 0, \quad \text{for any other } k \end{aligned} \tag{2.3}$$

The KL divergence is also discrete here, and so, unlike the integral version used in the VAE derivations, it will be the discrete version

$$D_{KL}(P(x)||Q(x)) = \sum_x \left(P(x) \cdot \log \left(\frac{P(x)}{Q(x)} \right) \right) \tag{2.4}$$

which, when applied to the quantisation output, becomes

$$D_{KL}(q(z_q|x)||p(z_q)) = \sum_k \left(q(z_q|x) \cdot \log \left(\frac{q(z_q|x)}{p(z_q)} \right) \right) \tag{2.5}$$

Now we know from equation 2.3 that for each k, $q(z_q|x) = 0 \forall k$ except $\arg\min_j ||z_e(x) = e_j||_2$(closest codebook entry). So equation 2.5 resolves to

$$\begin{aligned} D_{KL}\big(q(z_q \mid x) \,\|\, p(z_q)\big) &= \log\left(\frac{1}{1/K}\right), \\ D_{KL}\big(q(z_q \mid x) \,\|\, p(z_q)\big) &= \log K \end{aligned} \tag{2.6}$$

This is a constant term. However, the prior shouldn't really be uniform because there will be dependencies in real-life scenarios. In other words, the appearance of one codeword is not independent of the appearance of other codewords. So finding the prior is usually done via ancestral sampling. The seminal paper's authors used PixelCNN, an autoregressive model, to do that for images and WaveNet for audio [21].

Now, the loss function that we can compute so far is the reconstruction loss, defined as the negative log-likelihood of the data under the decoder, conditioned on the quantised latent z_q:

$$\mathcal{L}(\theta, \phi, C\ ;\ x, z_q) = \underbrace{-\log p_\theta(x|z_q)}_{\text{Reconstruction Loss}} \tag{2.7}$$

It is important to note here that it is not an expectation, as the posterior is now deterministic, unlike the VAE. Now comes the most important part of the VQVAE, which is the propagation of gradients through a non-differentiable quantisation step. To do this, straight-through estimation is used, which passes gradients straight from $z_q(x)$ to $z_e(x)$. That is,

$$\frac{\partial \nabla \mathcal{L}}{\partial z_e} = \frac{\partial \nabla \mathcal{L}}{\partial z_q} \tag{2.8}$$

This does work; however, this prevents the codebooks from changing. To mitigate this issue, we can add a new term $||\text{sg}[z_e(x)] - e_k||_2^2$. Here, e_k is the chosen code which is "closest" to the encoder output. This is called codebook loss. The sg$[\cdot]$ operator is called stop-gradient, and it prevents the operand from being changed by gradients. This basically pulls the codeword towards the encoder output that chose it.

This brings another problem, however, as encoder outputs will drift wildly with this setup as the codes get closer. To prevent this, it is paired with another term that pulls the encoder outputs to the chosen codeword (an opposite pull). This is called commitment loss as it encourages an encoder output to commit to a single code, and is defined as $||\text{sg}[e_k] - z_e(x)||_2^2$. This is usually weighted by a metric β (usually 0.25) that makes sure this pull is not as strong. So, the total loss function for the VQ-VAE is

$$\mathcal{L}(\theta, \phi, C\ ;\ x, z_q) = \underbrace{-\log p_\theta(x|z_q)}_{\text{Reconstruction Loss}} + \underbrace{||\text{sg}[z_e(x)] - e||_2^2}_{\text{Codebook Loss}} + \underbrace{\beta \cdot ||\text{sg}[e] - z_e(x)||_2^2}_{\text{Commitment Loss}} \tag{2.9}$$

2.4.4 Relevance in HAR

Autoencoders are not relevant in the context of OPERANet nor HAR via CSI/Spectrogram information. This is because the latent space is deterministic, and it is a discriminative model. Downstream classification being the primary goal, autoencoders just can not be used for that purpose. That brings us to the other two alternatives, which have been explored to various degrees of success - VAEs and VQ-VAEs. This does not mean that Autoencoders find no use within the context, as they can be used for unsupervised denoising/compression of spectrograms or semi-supervised learning before a light classifier. However, they do lack any probabilistic modelling, so they are much weaker at handling multiple modalities or principled fusion.

VAEs, on the other hand, have been used extensively in the context of both the field as well as the dataset. It has been shown that Multimodal VAEs with a latent space fusion implementation can learn a shared latent space from multiple modalities of unlabelled data. It can also use this multi-modality to denoise and recover missing views and then do few-shot classification with acceptable levels of performance [15]. However, VAEs do still suffer from posterior collapse in the case of low-variance input signals. It can overfit to room and access point geometry and would require curated learning. They do make up for these shortcomings with reconstruction-aware features and the robustness in the way they handle missing data via multimodality.

VQ-VAEs have also been used on the OPERANet use case on the Wi-Fi CSI dataset. It was seen to have low reconstruction error, and judging from the latent space clustering, enabled compact features viable for HAR. It was designed to fuse multi-receiver CSI, and the applications can be extended to communications tasks as well since the model was able to effectively reconstruct noisy data and missing modalities via learned latent space representations [4]. The reasons for this are evidently due to the discrete representations. Quantisation forces the model to learn discrete, compact and discriminative codes that are representative of actions, which improves downstream classification, especially considering the reduced dimensionality.

However, VQ-VAE implementations are not without their challenges. It is hard to comprehend their inner workings as they contain two convolutional neural networks that feed off of each other. Losses and other metrics give an overview, but code usage and variety are of the utmost importance. To that effect, code perplexity is a metric that should be tracked and encouraged. It measures the effective number of codes used. It is 1 if all encoder outputs are assigned to 1 code and K if all the codes are equally used.

It is defined mathematically as the probability of code k being used.

$$P = \exp(-\sum_{k=1}^{K} p_k \log p_k) \tag{2.10}$$

Monitoring this metric would ensure that codebook under-use or collapse does not occur. There are multiple methods to ensure it, which shall be discussed later in the methodology and execution sections. But methodology notwithstanding, ensuring that a healthy variety of codes are used by the VQ-VAE ensures that the latent space will grow with a healthy enough backbone that allows us to viably use it for downstream classification. A low perplexity value is a definite red flag that tells us one of two things. The encoder output has collapsed so much so that its output subspace is in and around a single code, meaning that the output subspace is way too dense for meaningful quantisation to occur. Or, the quantisation step itself is not working as intended. The codebook could be frozen where maybe β is so high that the encoder output is over-penalised into a narrower output region. It could also be that the codebook's size and scale are too small to account for the variance in the encoder output z_e. All these contribute to the fact that perplexity will, throughout this study, remain one of the more important metrics under scrutiny.

Chapter 3

Methodology

This chapter is concerned with the methodology of the project, as implied by the title. But there is an intention to explain the reasoning as well, to facilitate the critical evaluation of methods used with contextual reasoning. In the case of a relatively novel approach such as this, the literature is still developing, and as such, there are no major norms in the way of what, how and when to do processes. So, most of these processes have been done with efficiency and ease of implementation in mind because of the limited time frame within which the project was implemented. The code and associated documents can be found in [10].

3.1 Dataset Overview

The dataset used for the project, as mentioned above, is the OPERANet dataset [3]. This project is concerned with the PWR part of the dataset. The reasoning for this choice is largely based on the context of the project itself. The dataset contains multiple sensor data - Wi-Fi CSI, Ultra Wide-Band (UWB), PWR and Kinect. This project focuses on PWR data. The authors have multiple other papers on the implementation of multi-modality and VAEs on CSI data as well, so there will be some exploration of that in the results and analysis section as well, for reference. Most of the conversation from now on involving the dataset would exclusively be talking about the PWR part of the OPERANet dataset.

The PWR dataset consists of data taken from 4 channels. One is a reference channel, and 3 are surveillance channels. Since the PWR system, by virtue of being passive, does not care for the content of the Wi-Fi signal or the kind of signal it is, it can use any third-party signal as the illuminator. However, since the measurement is taken relative to the reference signal, there is a need for a constant reference signal. Then the PWR system compares these two to determine relative range and Doppler frequency shift. The only missing argument is the measurement rate, but this depends on the processing power of the software component as well as the number of simultaneous channels to process. In this case, it is set as 10 Hz empirically [3].

3.1.1 PWR Dataset Split

Throughout the project, the experiments were conducted on two variations of the dataset.

- (200, 30) sized chunks of spectrogram data corresponding to 3 seconds of activity, which are contiguous in experiment number and activity tag.
- (224, 224) resized and normalised spectrogram data from pipelines that were used to conduct the research in [4].

The reasoning for this is to compare the results on an "image" made from cutting up contiguous blocks of experiment data into 3 seconds of activity and using a preprocessed normalised pipeline that resizes these same "images" for effective representation.

The process by which the preprocessing has been done will be described in the coming segment for use in the justification of its creation. Figure 3.1 shows the lengths of each experiment. The experiment numbers do correspond to some layout changes and such, but generally speaking, they do average around

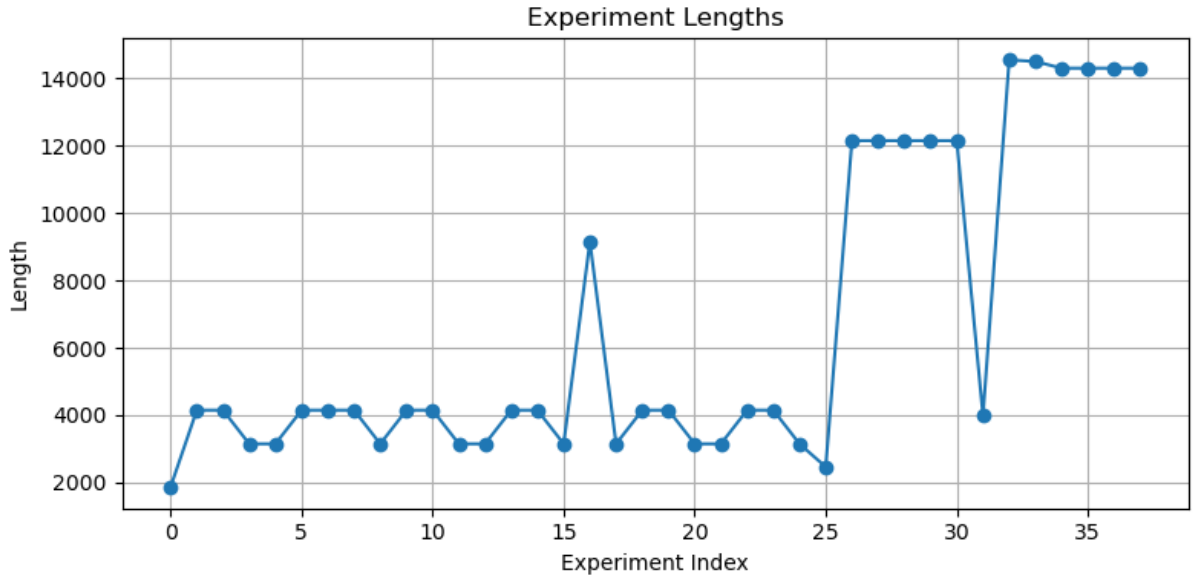


Figure 3.1: Experiment Lengths from the PWR dataset

Activity	Count
noactivity	150278
walk	37308
bodyrotate	30160
liedown	8349
standfromlie	8173
sit	7903
stand	7730

Table 3.1: Activity frame counts in the entire PWR dataset

the 4000 frames area. This is a bit of a concern, strictly speaking, as some of the less prolific activities do only number in around 7000 (Table 3.1). However, there is a more pressing concern to only feed inputs that are contiguous here because of how the dataset experiments are done. There is a need to make sure that the inputs are contiguous blocks of experiments and activities.

As seen in Figure 3.2, there are some issues with this arrangement as the vast majority of data falls in the `noactivity` tag, and it seems as if a lot of it is of different sizes, going from the longest contiguous block to the smallest one (Figure 3.3). From these plots, it is easy to understand that there are two problems here that need addressing in the dataset before processing.

- There is a definite need to skip the `noactivity` tag - it is extremely large and would only serve to clog the latent space, nor would it give any viable data, as in the context of HAR, it is rather irrelevant to know when somebody is not doing anything. It is also intuitive that it will have considerable bleed into other activities like `sit`, and `stand` - namely, static activities.

- There is a need to figure out a viable chunk length. A length that is as long as possible to get enough frames in a specific chunk and still be short enough that there is a viable amount of data from each label that can be used to train the VQ-VAE.

So there was a need to figure out the ideal length of the input. Since the heights are fixed at 200, it was worth identifying the average length of a frame and how many frames would make up, say, a 3-second window, done via the timestamps. This calculation yields an average of 30.16 frames for a 3-second window across the dataset. Then, figuring out how many equally sized chunks could be extracted from the entire dataset, making sure that each chunk had the same experiment number, activity, and length, was done via an exploration of the entire dataset by chunk number. The results are available in Figure 3.4. The chunk size 30, which is a 3-second window, corresponds to the elbow region of the graph and thus can be put in the viable zone before the benefits of reducing the size flatten out. As such, the

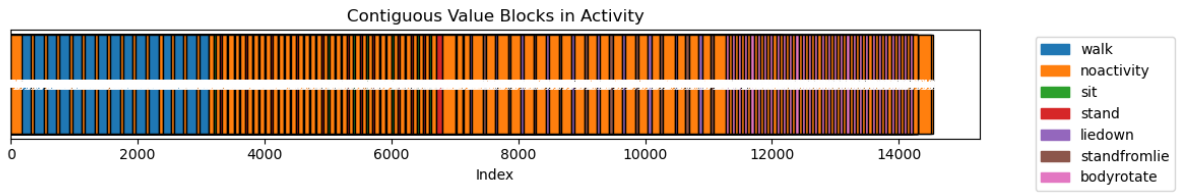


Figure 3.2: Contiguous blocks in PWR

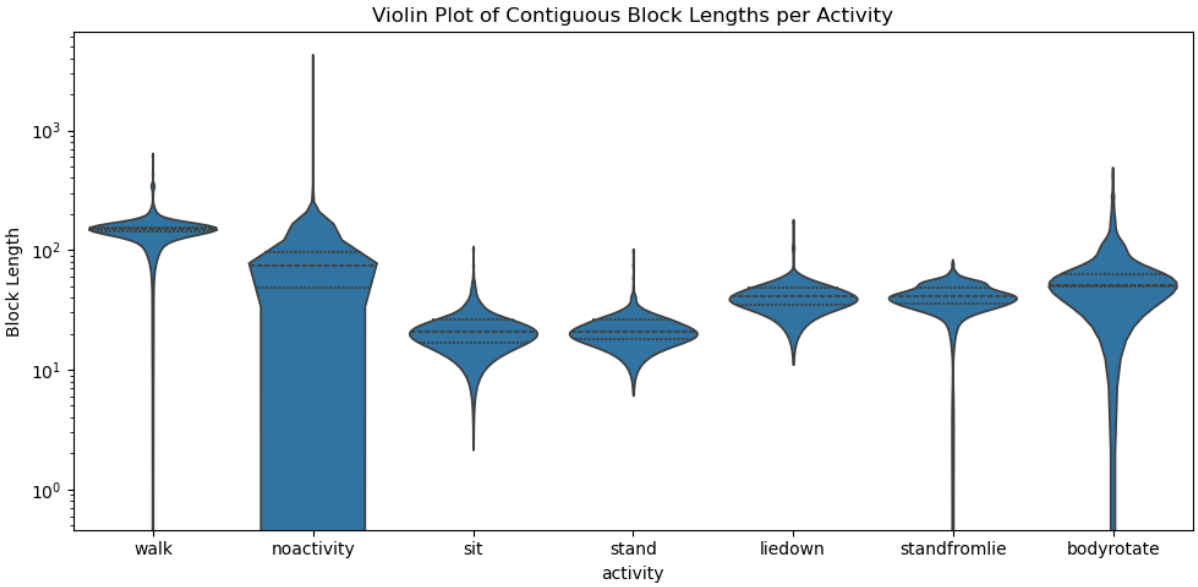


Figure 3.3: Violin plots showing the length of each contiguous block of activity

dataset was constructed to consist of "images" that are (200,30) and then used to train the VQ-VAE and extract and compare results. For both implementations, the train-val-test split was consistent using the `scikit-learn` package's `train_test_split` function with `test_size`=0.2. That is, 20% of the dataset is for testing.

3.1.2 Dataset Preparation for ResNet18

However, this is only in the case of VQ-VAE. As mentioned above, there was a ResNet18 benchmark supervised learning test done as well during the course of the project, mainly to ascertain the extent to which supervised learning could carry out the classification. For this specific purpose, the images were not made into equal-length chunks. Contiguous blocks of activities were chosen, stacked into an RGB format (since there are 3 channels of data, this almost corresponds to what a ResNet model would look for), and resized into (224, 224) images, which were then saved and fed into the ResNet18 model. Examples of the created images are given in Figure 3.5.

3.2 System Overview

This section is supposed to give an overview of the proposed system (both the ResNet18 implementation as well as the VQ-VAE implementation) before a deep dive into the implementation details in the later sections.

3.2.1 ResNet18 Implementation

The ResNet18 implementation was done with implementation efficiency in mind. As explained in section 2.3, supervised learning methods are harder to generalise and suffer from data hunger issues. So this implementation is in no way a result-oriented implementation and just a demonstration to showcase that the multi-channel PWR dataset can be classified with a fair amount of accuracy, given the right kind

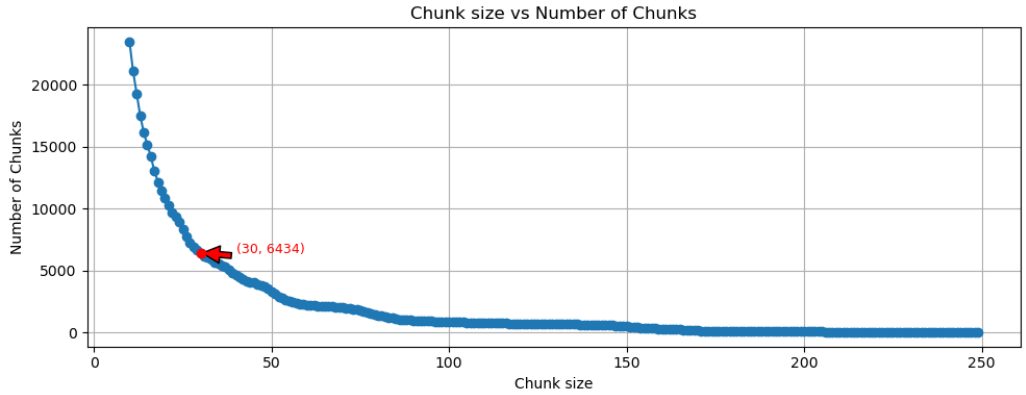
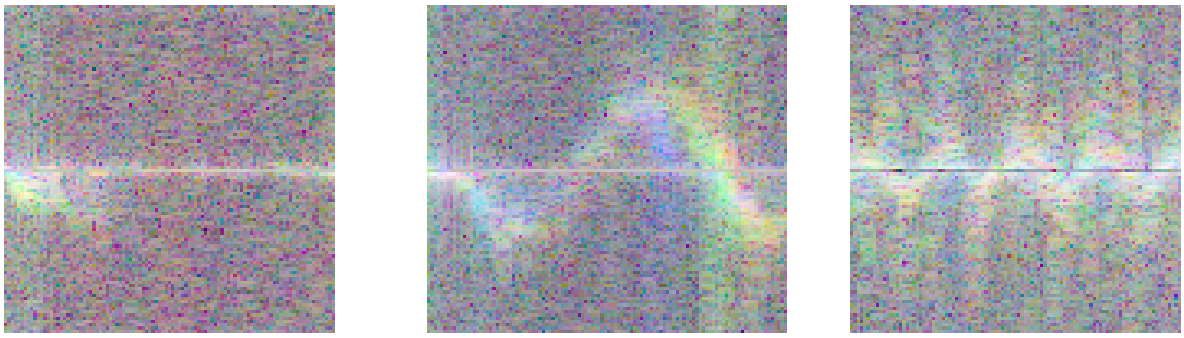


Figure 3.4: Chunk Size to Number of Chunks



(a) Liedown (b) Walk (c) Bodyrotate

Figure 3.5: Example images from the ResNet18 input

of machine learning inputs. The architecture of the base ResNet18 model is described in [8]. This was made for image classification in general and thus is expected to perform extremely well on the image data provided. In this situation, the ResNet18 architecture has been implemented from scratch using `PyTorch` as a means of exploring that avenue of machine learning.

The original architecture of ResNet18 is given in Figure 3.6. This architecture was simulated via a custom `PyTorch` implementation and trained on the resized contiguous blocks for 15 epochs. There were no real hyperparameters that were of interest to this implementation specifically, since most of the layers were pre-determined. We resized all the inputs to the same input size, and the learning rate was set at 0.001 as per norms.

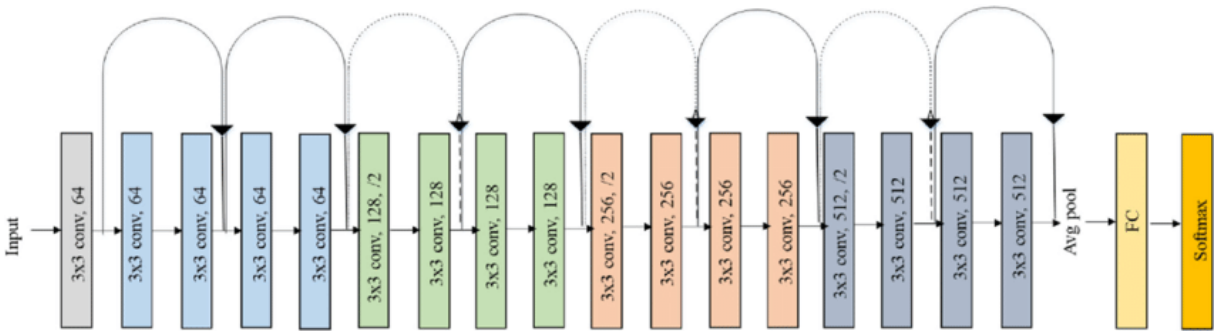
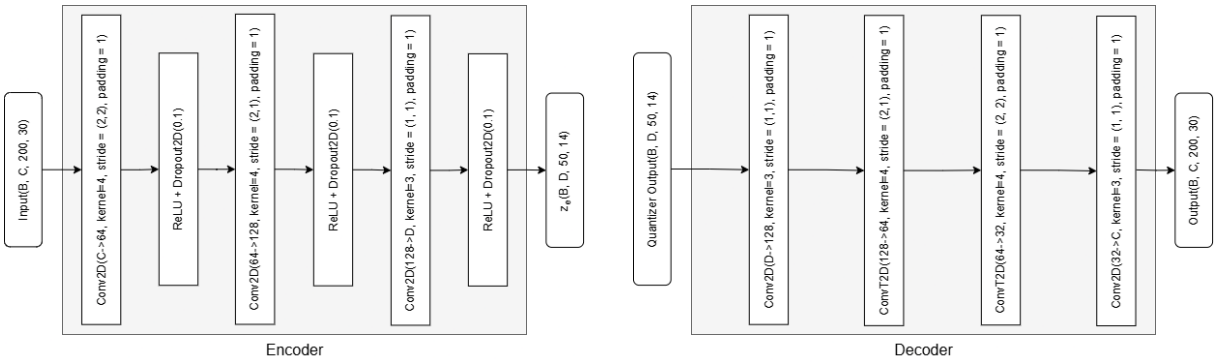


Figure 3.6: Original ResNet18 Architecture. Image sourced from [17] based on [8]



(a) Encoder architecture (b) Decoder architecture

Figure 3.7: Single Encoder-Decoder VQ-VAE Architecture

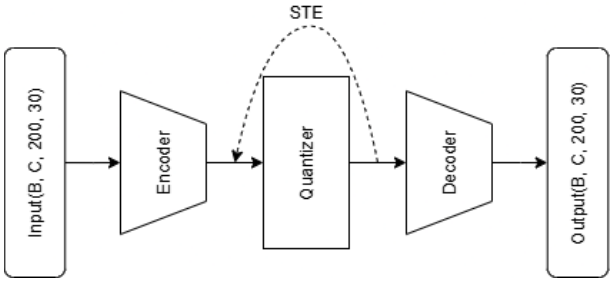


Figure 3.8: Full VQ-VAE architecture for the single encoder-decoder setup

3.2.2 Single Modal VQ-VAE Implementation

The first implementation that was explored on the "chunks" of contiguous experiments and activity as defined in Section 3.1.1 is the single modal VQ-VAE implementation. The idea was to create a simple encoder-quantizer-decoder architecture that could take in all 3 channels of data as channels within the CNN architecture of the VQ-VAE itself. This was done to explore how viable this is for the dataset. The presumption being that such integration of multimodality, called early fusion (or input fusion), would not give as bright results as the late fusion approach, where we would have multiple encoders for multiple modalities of data, the modalities here represented by the multiple surveillance channels.

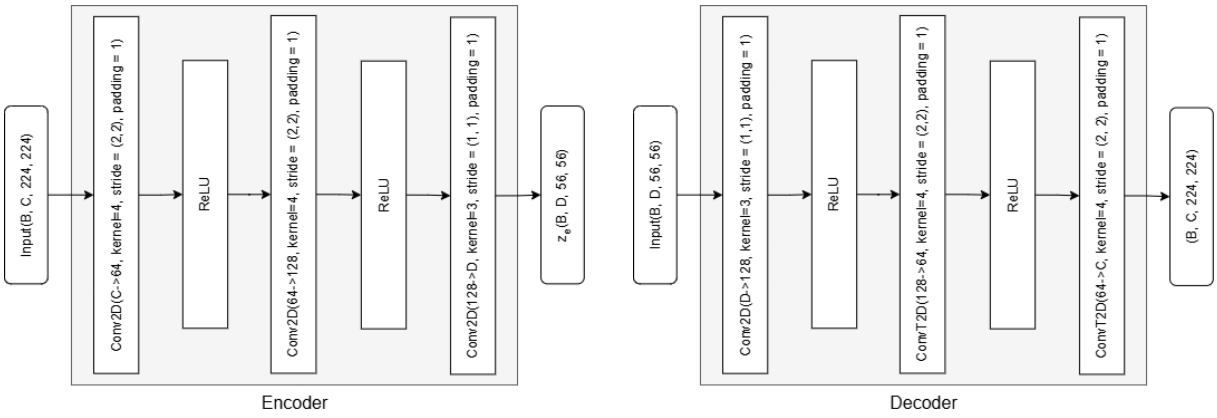
This is largely due to the size of the dataset available and how multi-channel inputs enlarge the size of the data. For a VQ-VAE setup, this would mean trying to generalise across three times as big a feature space with the same amount of data. This also means a lot of tweaking of hyperparameters to prevent codebook collapse. This could only be done experimentally, as it is borderline impractical to try and predict how codebook perplexity will evolve during training of the system. However, codebook collapse was observed to occur very early into the training and could be diagnosed as early as the 5th or 10th epoch, saving a lot of time in diagnosis.

The Straight Through Estimator (STE) in Figure 3.8 means that gradients are copied and passed through the quantisation step without affecting the differentiability of the entire setup. In the context of the VQ-VAE, for both implementations, it means that on the backpropagation step, the encoder output z_e is quantised to the nearest z_q and then $z_q \leftarrow z_e + \text{sg}(z_q - z_e)$, where sg is the stop gradient operator. This allows the encoder to learn even through the non-differentiable quantisation step and is a cornerstone in the VQ-VAE implementation.

The architectures for the encoder, the decoder and the full versions are given in Figures 3.7 and 3.8.

3.2.3 Multi-Modal VQ-VAE Implementation

The multi-modal VQ-VAE implementation follows the same principles as the previous one. However, the dataset used is a normalised version whose input makes it (224, 224) instead. The pre-processing emulates the studies conducted in [4] and is intended to prove to have better readability since the preprocessed spectrograms are plotted better. The base architecture is practically the same as the single-modal VQ-VAE implementation. Except, in this case, instead of having one CNN take in all 3 modalities as 3



(a) Encoder architecture (b) Decoder architecture

Figure 3.9: Multi-Encoder-Decoder VQ-VAE Architecture

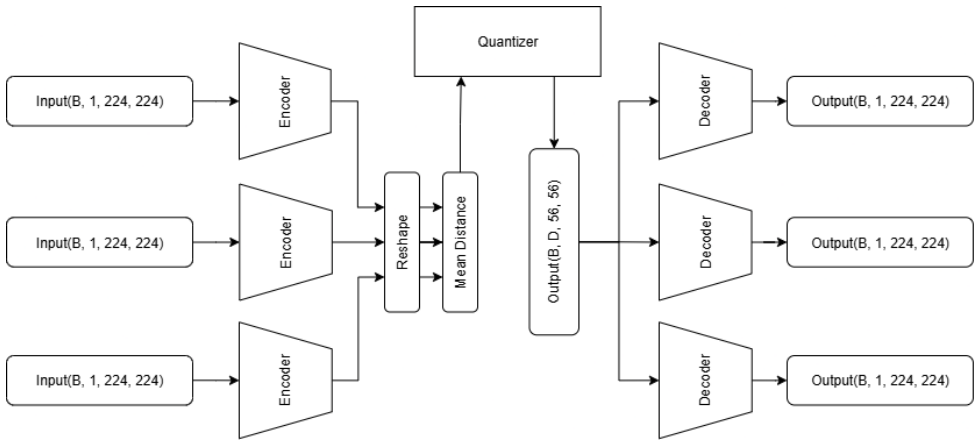


Figure 3.10: Full VQ-VAE architecture for the multi-encoder-decoder setup

input channels, we have 3 encoders (and 3 decoders) that take in each modality and work on them separately. This obviously creates some issues with the quantizer, as now multiple z_es require distances to be calculated and codes to be assigned independently.

The most elegant and simple solution is to combine the distances by taking the mean of all three distances and finding the closest code and using that one code to quantise the input [4]. This is paired with some more tweaking in the encoder and decoder, mainly to reduce their complexities because the training time and RAM usage go up very quickly per channel in this implementation. However, this is to be expected as the number of CNNs in the entire architecture jumps up from two to six. The architectures are given in Figures 3.9 and 3.10.

3.2.4 Implementation Details

Most of the code has been uploaded to GitHub, accessible via [10]. The implementations have been coded in using Python 3.12 and run in the latest versions of `PyTorch+CUDA` available. Most, if not all, of the implementations were tested on a local machine with an NVIDIA RTX 3060 6 GB mobile VRAM GPU, an Intel i7-10870H CPU, and 32 GB of RAM. However, this proved insufficient to run the bigger VQ-VAE implementations, and as such, Google Colab's free T4 GPU environments were used for the majority of the results that will be shown in later sections. However, Colab free does have runtime limitations around 2.5 hours of one task and that has limited the epochs to 200 instead of the ideal 500 for the multi-encoder setup.

3.2.5 Hyperparameters

This section covers most of the hyperparameters and their relevance, and the decisions that came with the current implementations. There is no need for hyperparameter exploration with the ResNet implementation since that is not the primary study target, nor are there any more relevant hyperparameters than the learning rate and the epoch number.

Number of Embeddings(K)

The number of embeddings (or K) is the number of codebook vectors available for the VQ-VAE for quantisation. It is also called the size of the codebook, though that description leaves a bit more to be desired. It is important to notice that this does not mean the codebook has more options. K increasing gives the quantizer more discrete capacity, that is, it can capture finer details. However, this does increase the risk of under-use of codes, code collapse and a proportional use of memory. This parameter has been set to 128, 256 and 512 as per most papers discussed throughout the project [4].

Embedding Dimension(D)

This is the channel dimension of each code and vector, and the latent space. This is basically the depth of the latent space. The higher D is, the higher the representational power. However, this comes with a much higher decoder performance requirement. This also influences the data hunger of the entire system. A higher D will also mean that the bottlenecks are not as effective, since they are closer to the input dimensions and, as such, reduce the effect of condensing the input space into a lower-dimensional space and retaining the important parts of the data. Too high a value can also slow training by a lot and hurt the utilisation of codes, reducing perplexity, which is generally a bad thing, as this means a worse clustering algorithm on the latent space.

Beta(β)

β from Equation 2.9 is the weight by which commitment loss is scaled. So it can be referred to as the commitment weight, though there is no standard way of referring to it. This balances the commitment loss against reconstruction loss. It would do well to remember that commitment loss is intended to push encoder outputs closer to their chosen codes. Too high of this would mean that encoder outputs would violently try to stick to their first chosen code and degrade reconstruction. Meanwhile, too low a value would mean codes drift too much and get ignored. With this information, it is easy to see that since early epochs would see very random code assignments due to vastly nonsensical encoder outputs, commitment loss should be much less of a concern, which is not the case later down the line, when it should be more of a factor in deciding gradients. So, a ramping β was hypothesised and then later proven to grant better performance [12]. So the idea is to ramp the β from very low, slowly up to the ideal value to prevent noisy early outputs from scrambling the quantisation. The ideal values for most applications and datasets are around 0.25, which is also the one used for the seminal paper that introduced VQ-VAE. In this project, this value has given us viable results as well [21].

Latent Grid Size

This is a hyperparameter that is not actually set via an argument passed to the VQ-VAE class, but something that is set via the strides and padding in the encoder and decoder convolution layers (mostly the encoder). In Figures 3.7 and 3.9, it can be seen that they are (50, 14) and (56, 56) respectively. That is, in the single CNN structure, the latent space is reduced 4 times in one dimension and 2 times in the other. This is because of how small the "width" is, which is 30. In the multi-CNN structure, this is more symmetrical in both dimensions by being reduced 4 times. This metric determines the number of tokens per "image". Fewer tokens mean that computation is cheaper. However, the issue that arises is a more severe capacity issue. The more you compress a feature space, the less informative it gets, and there is a specific point at wherein the latent space loses more info than it should to represent the feature space informatively enough.

Reconstruction Head

The chosen kind of loss metric is pretty important here. There are two choices: L1 loss or MSE. MSE (or L2) minimizes $\frac{1}{N}\sum(x-\hat{x})^2$. L1 loss minimizes $\frac{1}{N}\sum|x-\hat{x}|$. This means that MSE gradients scale

quadratically. That is, errors are fixed much faster and smaller errors are fixed more slowly and smoothly. However, this also means that there might be an outlier issue, and models tend to average over targets. L1 gradient, on the other hand, treats errors much smoothly and as such is more robust to outliers and preserves some edges better. However, its convergence is less than zero due to the flatness near zero gradient. In most VQ-VAE implementations, MSE is the preferred loss due to the normalised images and the Gaussian nature of most normal noise.

3.2.6 Training Parameters

Learning Rate and Batch Size

Learning rate can be shared, or it is possible to give the codebook a slightly higher learning rate than the encoder/decoder setup to encourage utilisation. However, in this study, the choice is to keep the shared learning rate since such tuning would require more analysis over the entire training parameters subspace, and that is a bit out of scope for the amount of time allotted to the study itself. The batch size has been chosen to be 64. This is not a completely inconsequential metric. A smaller batch size increases the variance of all the losses as we are using Naive Monte-Carlo. So this metric was chosen experimentally.

Changing Values

Some values change over the epochs that are very important to the intrinsic workings of the training. One of the more important ones is β for reasons mentioned in Section 3.2.5. This is reflected in the way the training is implemented by β starting from 0.01 and ramping up to the final value of 0.25 over a set `warmup_epochs` (set to 60). The training epochs were set to 200. The other example under consideration of an implementation of VQ-VAE [4] uses 500 epochs. However, due to technical limitations, this study has chosen to keep it to 200 epochs to still receive viable results that are malleable enough to tweak settings and still be informative enough to inform later decisions.

Quantization

The `warmup_epochs` mentioned in the above section also serves to turn off quantisation and the beta ramp for that many epochs. The advantages of doing this are multi-fold. This would allow the encoder to bootstrap itself. A hard assignment via distance calculations and STE would give noisy, arbitrary assignments in the beginning. Recovery might be harder or impossible from these beginning assignments, depending on the β assigned. Giving the encoder setup some non-quantised steps gives some time for the weights to stabilise into a more sensible version. This, followed by a β ramp, would give the codebook clean assignments and higher perplexity. This could also reduce the loss spikes due to early assignment and would allow optimisation to be smoother.

3.2.7 Validation

The layout of the latent space has been explored via running the samples through the encoder for z_e and encoder+quantizer for z_q vectors. This latent space has then been reduced in multiple ways for the exploration of the general layout of the latent space as well. The most common being PCA→t-SNE(or UMAP). This was an exercise in showcasing how the latent space clustered intrinsically via unsupervised learning. The graphs are made with multiple options like z_e or z_q, pooling (channelwise mean or mean+std), reducer (t-SNE or UMAP), initialiser for t-SNE (random, PCA or noPCA), number of neighbours, minimum distance, metric (euclidean or cosine) for UMAP. This method, although it gives us an extremely large result space, allows us to see trends against multiple reducer parameters and arguments.

Finally, simple KNN cross-validation was run on latent features (z_q and z_e mean/mean+std or code-histogram features). PCA, when used, was fit within each fold to give a general idea of classifier viability. We did not evaluate KNN on t-SNE/UMAP embeddings, which are for visualisation and distort distances. Instead, stratified CV with a small grid over $k \in \{5, 10, 15, 30, 45, 50\}$ was used to pick the neighbourhood size. For histogram features, the Hellinger transform, which is well-behaved for distributions and aligned with the square-root trick—was applied [1]: for each sample, count codebook indices to form h, normalise to $p = h/\sum h$, then map to $\phi(p) = \sqrt{p}$ so that $d_H(p, q) = \frac{1}{\sqrt{2}}\|\sqrt{p} - \sqrt{q}\|_2$ equals the Euclidean distance between the transformed vectors. Visualisations (t-SNE) are shown separately to compare samples in latent space.

Chapter 4

Results and Analysis

This chapter is intended to give a full breakdown of the multiple results that were acquired through the implementation of the methods explained in Chapter 3. Most of these results are labelled and broken down with caveats and possible explanations of their current state.

4.1 ResNet18 Results

The ResNet18 implementation results are given in Figure 4.1. It is obviously a very good confusion matrix, even including the `noactivity` tag, which dilutes most of the dataset. However, as discussed already, this only means that classification is viable in the dataset. A supervised model will not be able to account for even minor changes in the experimental setup, and that means the replacement of Wi-Fi access points. In a real-life scenario, this would mean that the model would have to be trained on labelled data every time the router is displaced, which is impossible.

4.2 Single Encoder-Decoder VQ-VAE Training Graphs

The single encoder-decoder VQ-VAE has two implementations as explained above. The single-channel input, where only one modality is considered (via a single channel), and the multi-channel implementation, where each of the 3 modalities is fed as individual channels into the VQ-VAE. The training graphs for both of these implementations are given in Figure 4.2. The initial peaking from all the curves is because of the skipped quantisation step. We can see that reconstruction loss, the primary contributor to the loss function, is decreasing steadily as expected in both setups. There is something to be said about the total

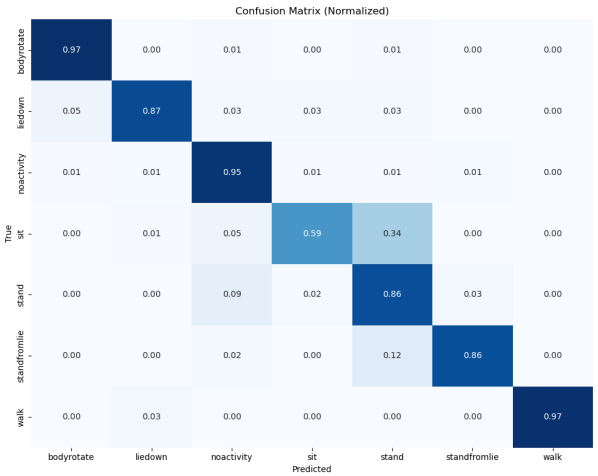
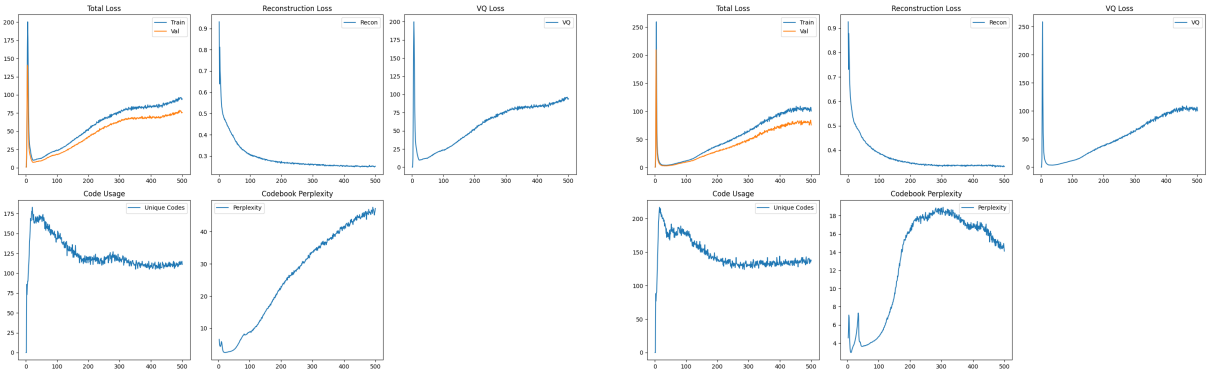


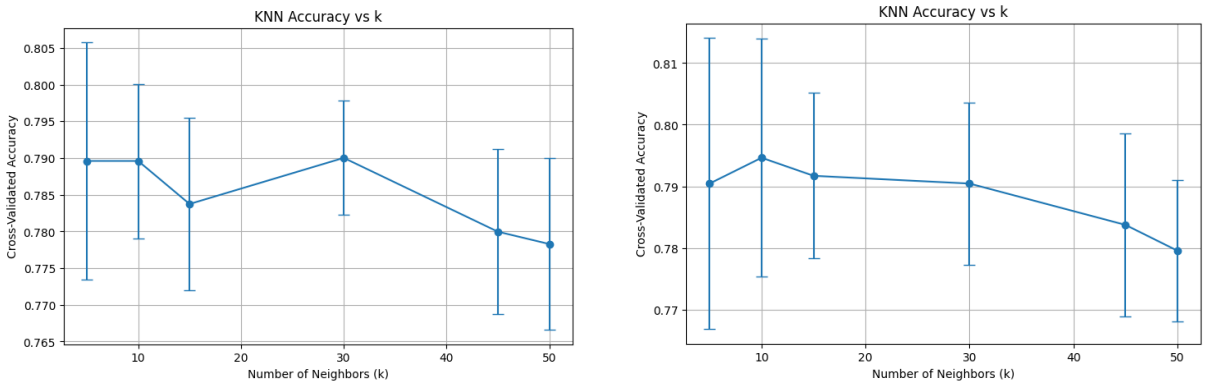
Figure 4.1: ResNet18 output confusion matrix



(a) VQ-VAE with a single channel

(b) VQ-VAE with three channels

Figure 4.2: Training graphs for single encoder/decoder VQ-VAE(K=256, D=64, $\beta = 0.1$)



(a) VQ-VAE with a single channel

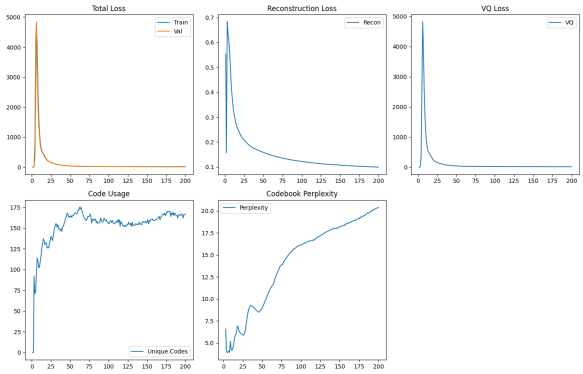
(b) VQ-VAE with three channels

Figure 4.3: KNN for single encoder/decoder VQ-VAE(K=256, D=64, $\beta = 0.1$)

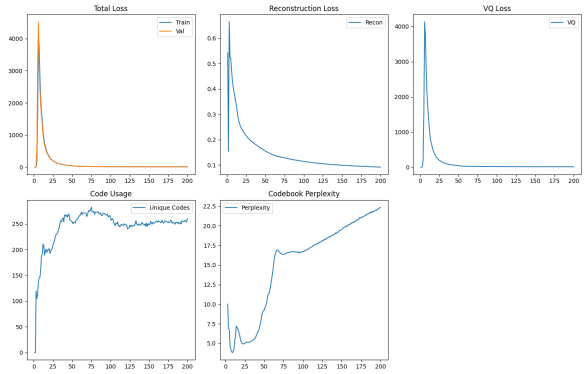
loss rising steadily, even though the reconstruction loss is decreasing, and that is because of the increase in VQ Loss that is seen across both. This, in turn, is because of the beta ramp and how it increasingly penalises differences between z_e and z_q as time goes on.

Another interesting observation is the perplexity. Curiously, single-channel perplexity rises and spreads much more uniformly, while that is not the case in multi-channel. It seems as if the perplexity plateaus way earlier and way less optimally than the single-channel version. This indicates that feeding multiple modalities via channels into a single encoder/decoder setup is not as viable as it may seem at the onset. The perplexity is significantly lower (45 in single-channel vs 18 in multi-channel), so there is a very clear under-utilisation of the codebook. This can be explained because both of these implementations have the same codebook size and depth. This means that the bottlenecked latent space is the same, but suddenly the setup has to deal with thrice the amount of information, which is causing a collapse. Unlike the single channel, the β ramp would cause some codes that are popular to attract closer vectors that reinforce perplexity loss. So, the hypothesis would be that a higher codebook size and dimensionality would, hypothetically, allow for an overall better performance. However, this should also be tempered as since the dataset size is fixed, it might lead to underutilization of codes instead. So some tweaking of β, `warmup_epochs`, `ramp_epochs` and other metrics would be necessary to offset such behaviour.

Regardless, the simple KNN algorithm results can be seen in Figure 4.3. Both models plateau around 0.78-0.80 Cross Validation (CV) accuracy, and there are only minute differences between the two. The error bars are very high on k=5, which suggests extremely noisy classification. However, it can be seen that with such viability of KNN, the latent space may have some usable local structure. However, they are not strongly separated, and as such, there is a headroom for a stronger method like linear, Support Vector Machine (SVM) or Multi-Layer Perceptron (MLP).



(a) VQ-VAE with K=256, D=128



(b) VQ-VAE with K=512, D=128

Figure 4.4: Training graphs for multi encoder/decoder VQ-VAE

4.3 Multi Encoder-Decoder VQ-VAE Training Graphs

The multi-encoder/decoder setup described in Section 3.2.3 was run for only 200 epochs. This is 40% of the number of epochs the single encoder/decoder setup was run for. The reasons for this have been mentioned already in the aforementioned section. Two major implementations for this have been tested, both of whose training graphs are given in Figure 4.4. The first noticeable difference is the stability of the algorithm. Total loss is noticeably more stable here, even though this implementation still has the β ramp and the `warmup_epoch` shock in code usage. In both settings, all losses crash around 20-30 epochs and flatten out, which is a sign of stable model performance. The code usage is around 170 ($\sim 60\%$) for the 256 code version and $\approx 255(\sim 50\%)$ for the 512 code version. This is more or less expected, as a bigger codebook lets the model explore more codes in the beginning.

Perplexity for both implementations could be improved, as they are both around the 20 mark. However, it is important to note that this is very much an early look into the model behaviour. The perplexity curves in both models are steadily increasing, and both models seem to be nowhere near the perplexity plateau range. Another good visual is how VQ Loss seems to approach zero and is consistent. This, unlike the previous model, suggests that outputs are very close to their chosen codes, β is not dominating its measures, and the codebook and encoders are adapting quickly and at the right pace.

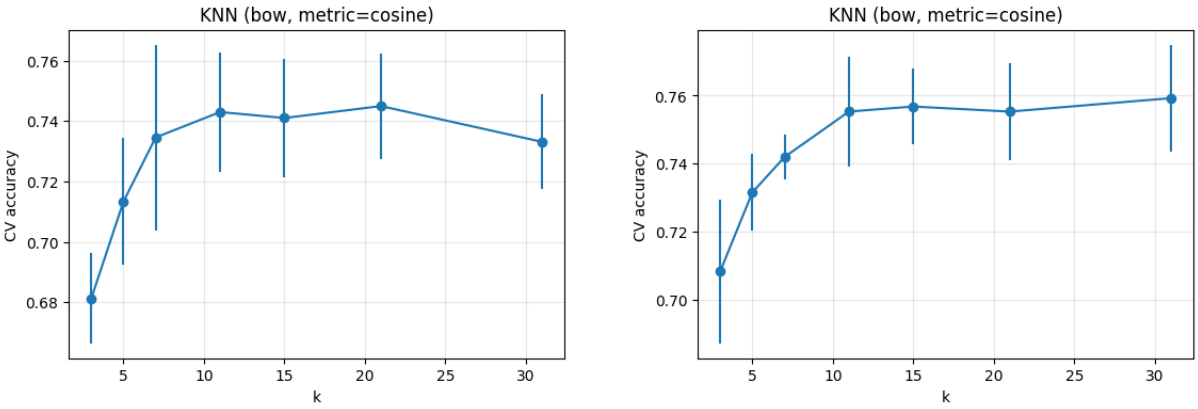
The multi-encoder/decoder architecture seems to be much better at creating clean reconstructions and a stable discrete mapping. However, the perplexity curve is slightly slow to catch up to the good trends that are observable across the loss functions. Further training until the perplexity plateaus would be a viable idea to figure out if the model can perform better. The KNN viabilities are given in Figure 4.5. There is a slight difference in that these KNNs were implemented on the Bag of Words histograms with Hellinger transformations on top of them, but it was observed that they were both viable and were within error values of ranges.

The observations are clearly consistent across the results. Single channel does have a slight edge over multi-modality implementation on the KNN viability. However, the multi-modality seems to clearly be more stable and shows considerable improvements in 40% of the epochs. This is consistent with the idea that code variety and usage dictate the health of the latent space and thus support a more clustered latent space. A viable hypothesis at this point is that a fully trained implementation of the multi-modal VQ-VAE will be both stable and more viable in terms of latent space representations.

4.4 Single Encoder-Decoder VQ-VAE Latent Space

The single encoder/decoder VQ-VAE latent space has been thoroughly explored as per Section 3.2.7 via Google Colab due to the relative ease with which training 500 epochs happens. So the grids shown in Figures 4.6 and 4.7 are the full extent of latent space exploration. There are a lot of options to see here, and it is not really clear what is going on without some exploration, so this section is intended to summarise most, if not all, learnable information from these exploratory grids.

It can be seen that throughout both grids, there are some consistent, key differences in z_e and z_q. The quantised latent shows tighter ring-like structures and clusters much closer than z_e. This is the codebook enforcing local neighbourhoods, and that is expected, viable and is proof that the codebook quantisation



(a) VQ-VAE with K=256, D=128

(b) VQ-VAE with K=512, D=128

Figure 4.5: KNN results for multi encoder/decoder VQ-VAE

is effective in creating such clustering. Even though their codebooks are much more bottlenecked, this fully trained model shows that some clustering is still possible even with a single modality of data. There is definite global class-mixing, but that would require some linear probing to deal with, and that is visibly out of scope for this study.

It can also be noticed that the cleanest manifolds observed for single channel implementations are $z_q + \text{mean} + \text{std}$ (last row) over UMAP (last three columns)(Figure 4.6). Several classes form elongated arcs with thinner overlap, which aligns with the higher perplexity and code usage observed earlier during training. This also corresponds to the higher KNN values.

In the case of multi-channel, the clusters are tighter, but there is a lot more entanglement in the global space. There is a lot of overlap between similar action types - static or dynamic. Static postures overlap, and dynamic postures seem to draw arcs instead of clustered islands. The fix is to make the model use more codes and diversify enough to make use of all available features to pinpoint the correct diversifying features. A more robust training set via class-balancing would also do a lot to make sure the model has enough data to generalise.

4.5 Multi Encoder-Decoder VQ-VAE Latent Space

The multi-encoder setup required a bit more consideration for computation and RAM when plotting results due to the constraints imposed by Colab. As such, the idea of implementing a full t-SNE/UMAP grid, like in the single encoder/decoder setup, was actively discouraged by technical limitations. So here there is the z_e latent space (Figure 4.8), which shows continuous encoder features with per-channel mean and std. This has no discretisation and is basically the output of the encoder directly. Figure 4.9 shows the same pooling of test data as before, but after quantisation. This is meant to show how much quantisation affects the structure of the latent space. The Bag-of-Words(Hellinger) in Figure 4.10 shows each point as a code-histogram per sample (square rooted so that Euclidean≈Hellinger as per Section 3.2.7).

Some consistent patterns are observable throughout the study. z_q forms a clean manifold in both k settings. Its neighbours are noticeably tighter than z_e, meaning that the study's primary idea of quantisation affecting local structures and clustering meaningfully is satsifiably proven, even to a small extent (The reader is still reminded that this is a half-trained implementation, the possibility of perplexity going up further than this and allowing for much better clustering is high). The BoW maps can be seen to be producing multiple "islands" that are consistent with [4], but there is still considerable global overlap. This, however, can be confidently claimed to be a hyperparameter tuning and epoch increasing issue that can be viably tested in further settings.

Another interesting fact is that going from K=256 to K=512 does not seem to change much about the structure of the data. This is a concern as that means the model is not learning codes that are variable enough to enforce more structure, but this could also be since there is a limited amount of latent information to be had. Adding more latent dimensionality would dissolve the usage of the bottleneck and lead to worse clustering, as per most CNNs.

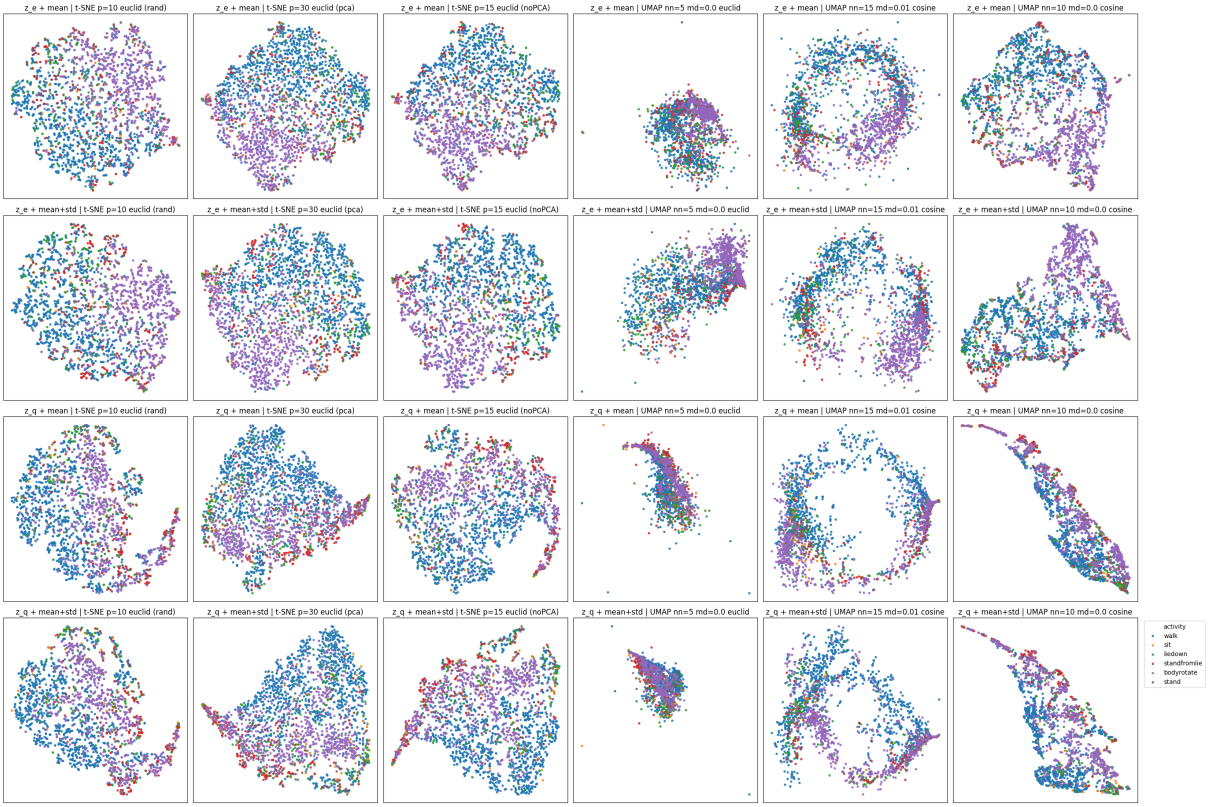


Figure 4.6: Single encoder/decoder single-channel VQ-VAE latent space

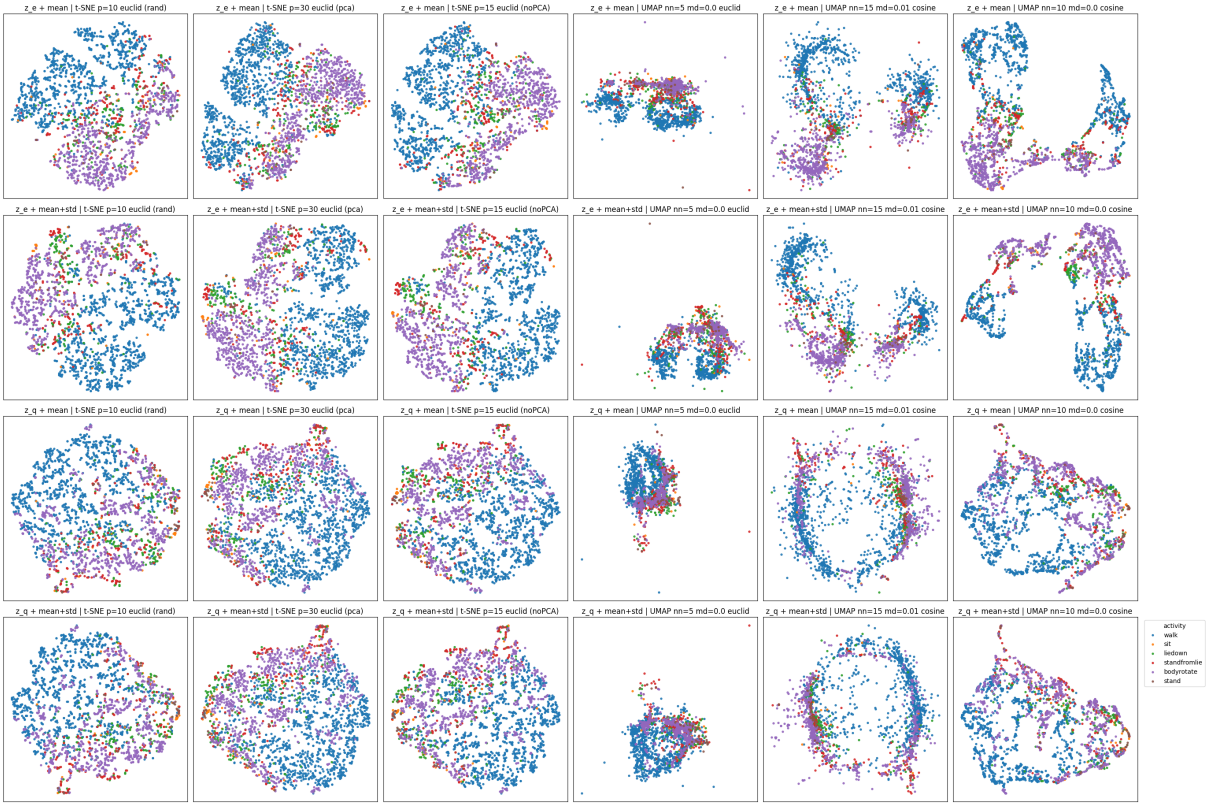
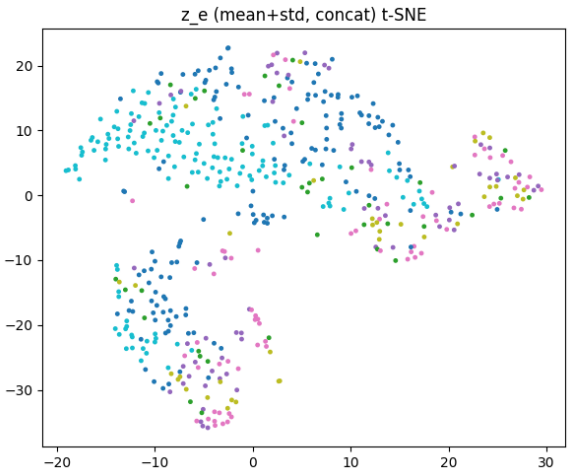
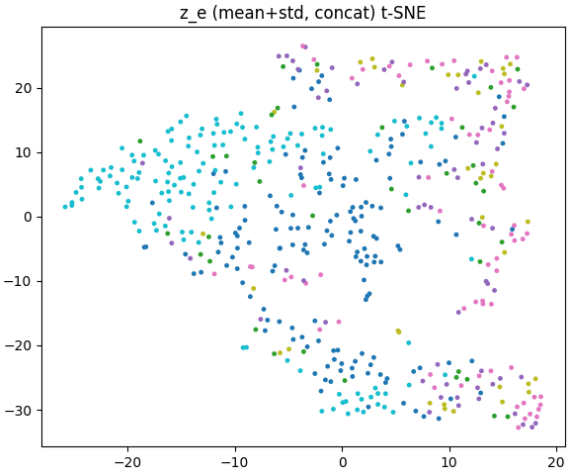


Figure 4.7: Multi encoder/decoder multi-channel VQ-VAE latent space

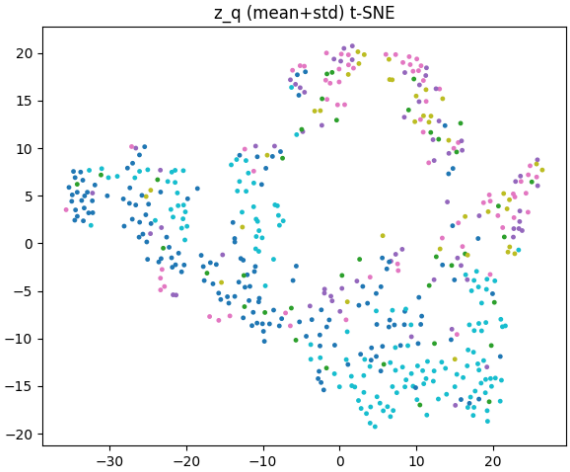


(a) z_e latent space for K=256

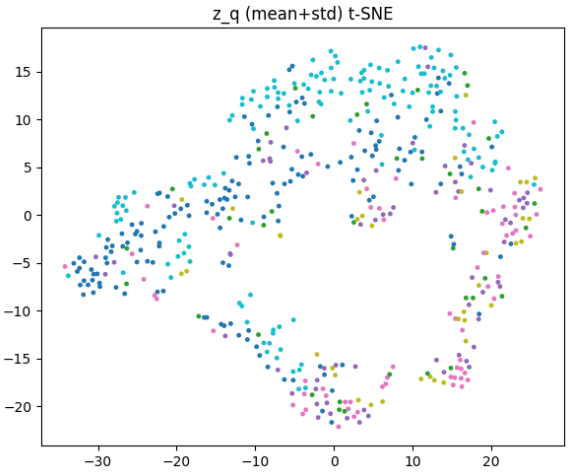


(b) z_e latent space for K=512

Figure 4.8: z_e latent space of multi encoder/decoder VQ-VAE

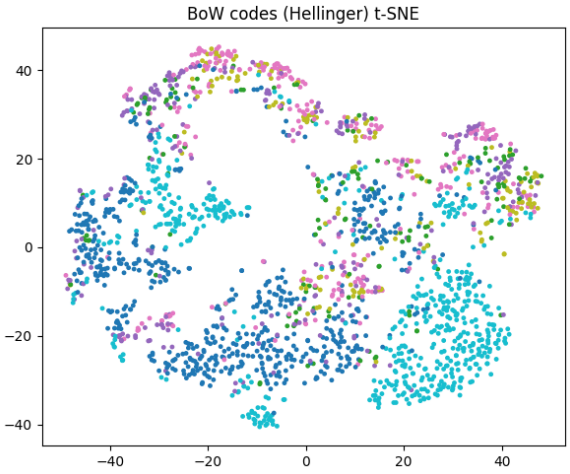


(a) z_q latent space for K=256

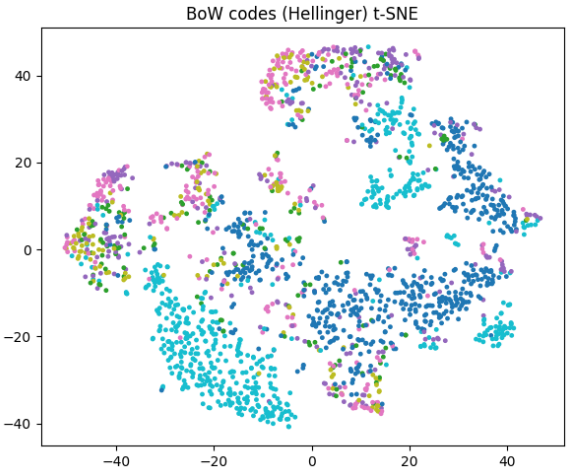


(b) z_q latent space for K=512

Figure 4.9: z_q latent space of multi encoder/decoder VQ-VAE



(a) BoW latent space for K=256



(b) BoW latent space for K=512

Figure 4.10: Bag-of-Words latent space of multi encoder/decoder VQ-VAE

4.6 Reconstruction Evaluation

Model Name	Codebook (K, D)	Reconstruction Error	Codes Used	Perplexity
VQ-VAE(1) with single channel	256, 64	0.2519	111	47.2
VQ-VAE(1) with multiple channels	256, 64	0.3326	138	14.1
VQ-VAE(2)	256, 128	0.1015	183	19.3
VQ-VAE(2)	512, 128	0.0905	279	27.3

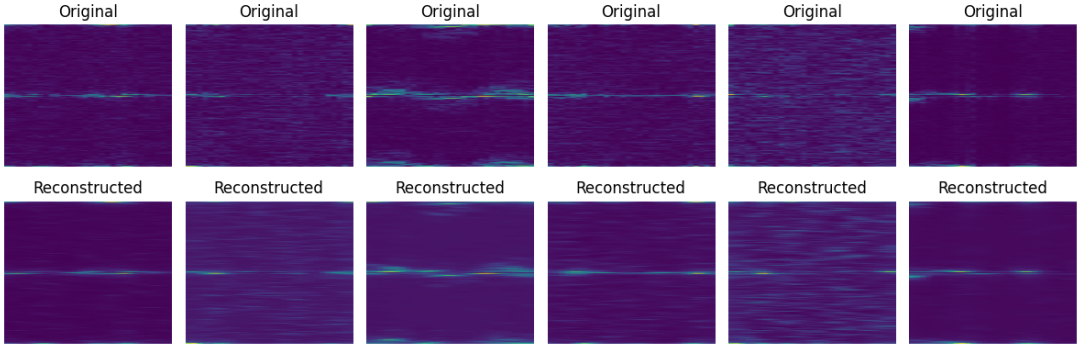
Table 4.1: Final tabulated results. VQ-VAE(1) refers to the single encoder/decoder implementation and VQ-VAE(2) to the multi encoder/decoder implementation.

Throughout the study, the main focus has been on latent space class separability, but it would be wise to review some reconstructions from the models to perhaps facilitate some scrutiny. The reconstructions are given in Figures 4.11 and 4.12. The figures are numbered 1-6 from left to right in all discussions below.

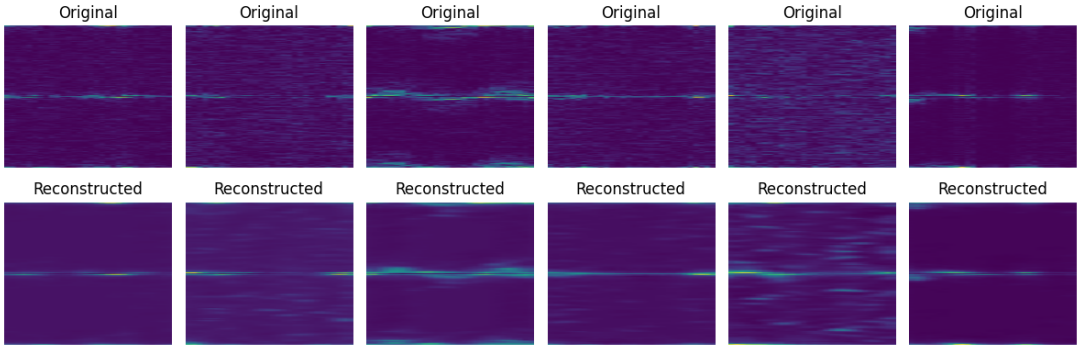
Mostly, the results show, as expected, very viable reconstructions throughout the board. However, on further scrutiny, some differences could be due to the technicalities of implementation. In Figure 4.11a, it can be seen that the single-channel tends to smooth out noise in a bad way. This is especially noticeable in images 2, 4, and 5, where noise can be seen to be blurred out, but still present in its entirety. This is most likely due to the absence of further modalities, making the model unable to distinguish between real and ambient signals. This lack of information or capacity to encode such information can also be seen in the 3rd image, where the sidebands have been smoothed out to a degrading degree. This shows a lack of capacity to preserve low-contrast features.

In Figure 4.11b, image 5 shows an interesting phenomenon, while in the single channel implementation, noise could not be removed; here, it has been removed, but bright speckled artefacts have appeared in place, likely due to the model trying to preserve finer detail while removing noise. Relative to single-channel, the three-channel variant preserves ridge continuity and sideband detail (images 2, 3, 4, and 6), while the single-channel variant more often blurs background noise but can thin out or break the central ridge (as in images 1 and 6). This suggests that added channels provide complementary cues that help the model distinguish structure from ambient clutter at the cost of retaining a bit more fine-grained details.

In Figure 4.12a, figures 0 and 1 look noticeably darker than the others, with parts of the ridge breaks it is blurry and noticeable. This does not seem to be the case in Figure 4.12b. In fact, even though the reconstructions seem to be doing well (as expected), the major differences seem to be that $K = 512$ seems to be giving sharper, brighter ridges with a bit more of the fine-grained details, while $K = 256$ smoothes the output more and loses some granularity in the sides. Both of the models seem to have some trouble denoising the input due to the low perplexity curves and the fact that it is not fully trained, but $K = 512$ does seem to be doing better in that regard (Reconstruction 2 noticeably).

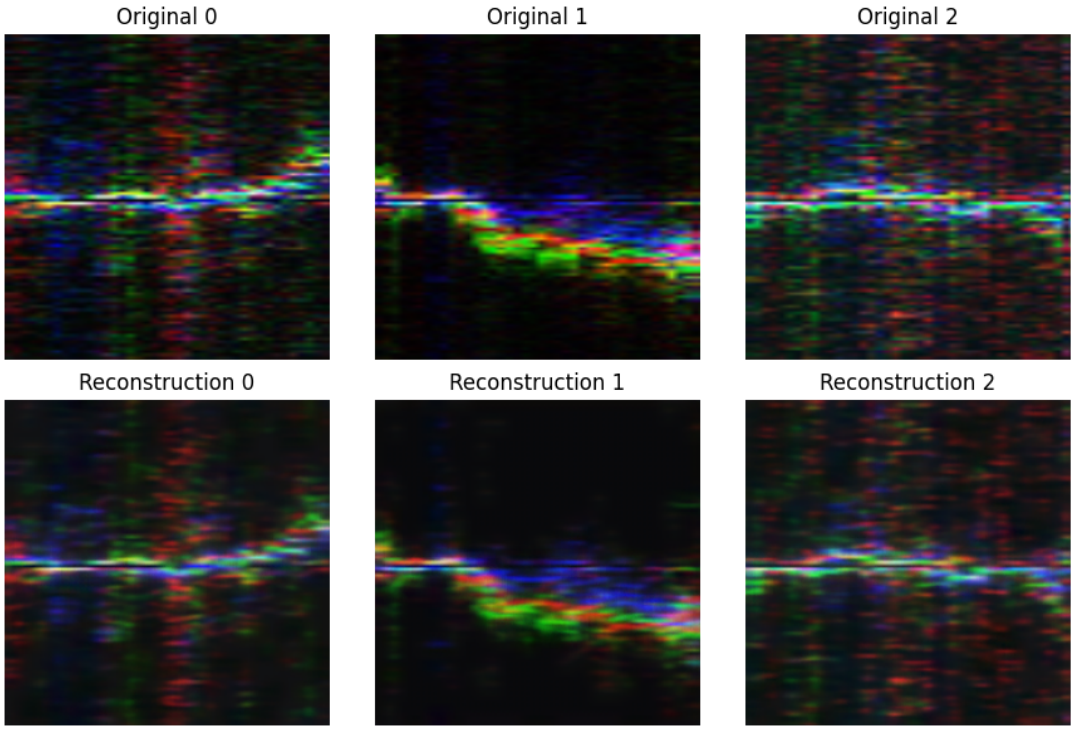


(a) Single channel VQ-VAE

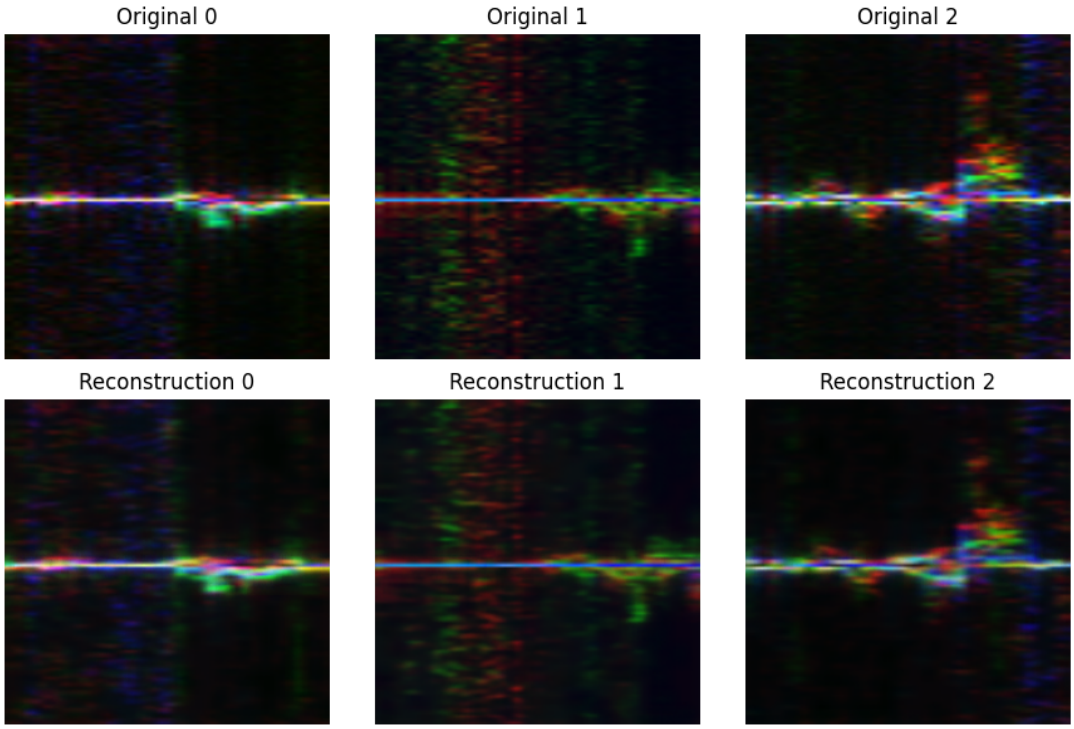


(b) Three channel VQ-VAE

Figure 4.11: Reconstruction examples of single encoder/decoder VQ-VAE



(a) K=256



(b) K=512

Figure 4.12: Reconstruction examples of multi encoder/decoder VQ-VAE

Chapter 5

Conclusion and Further Work

This is the concluding chapter focused on a summary of what learning outcomes have been achieved, what could have been fixed, and what the final propositions and conclusions are about what considerations need to be given for the systems and methods proposed to be viable.

5.1 Summary of Findings

To a large extent, the non-viability of supervised learning for HAR using PWR has been visibly demonstrated, both in this study, the citations and by the common logic that follows its implementations. The outcomes from the ResNet18 implementation, however, show that there is a very viable feature space in the PWR spectrogram data that can be used for HAR. If an untuned, unspecialized implementation of ResNet18 was able to achieve such accuracy over abstract RGB "images" of spectrogram data, it is understandable that the feature space contains viable inputs that can be used for HAR. However, due to the constraints of PWR, it falls upon unsupervised or few-shot learning techniques to achieve similar accuracy in results. To that effect, this study has focused on VQ-VAEs as a viable alternative.

It was also seen that multi-encoder/decoder VQ-VAE performs much better than a single-encoder/decoder setup when considering the multi-modality of data. And since multiple modalities of data are almost always indicative of more information and better classification, as demonstrated by the citations given, it is viably demonstrated that a multi-modal VQ-VAE is more stable, and better performance (because the curves aren't plateaued and the training epochs are 40% of the single enc/dec setup). The latent spaces from these implementations were plotted via dimensionality reduction techniques to showcase their spread. It was seen that there is definitely local clustering that gets dissolved in the global class spread. This was hypothesised to be due to the low perplexity values achieved during training, both due to lower epochs and lesser implementation time to do hyperparameter tuning.

The clustering in the latent spaces also showcased that quantised latents are clearly more clustered and show better class separability. This is a good thing as it shows that the discretisation provided by the quantisation step has potential in enforcing class structures in the latent space, supporting downstream classification. With further experimentation in this, there could be ways to maximise this phenomenon to give more weight to such behaviour. Even with little to no tuning, KNN was able to give competitive accuracy in the downstream classification task. This performance can only increase more with better linear probes. The research footprint for the entire process was computationally expensive during training, leading to some under-trained results in the multi-enc/dec setup, but once implemented, such encoders can process data to a latent space very quickly with GPU support.

5.2 Contributions

As far as contributions go, there is much accomplished in this study that are confirmations of hypotheses that can be made by looking at previous studies and the general behaviour of VQ-VAEs. However, this study has well-documented results that come from well-documented implementations that might be more viable to be looked at by an interested reader. A comparative study of single and multi-encoder/decoder VQ-VAE setups was explored, along with single-channel and multi-channel setups for

comparison. Their behaviour was documented for the purposes of assessing viability, and it was found that a multi-encoder/decoder setup with multiple modalities of data, trained more stably and gives more viable results than the others. Diagnostic plots for all these results are plotted, described and annotated for further reference. Baseline KNN was used to give the most basic implementation of a downstream classification method.

5.3 Limitations

The limitations are most notable in the technical aspects of the implementation. As mentioned multiple times throughout the study, the reader must notice the epoch count difference between the multi- and single-encoder/decoder setups when evaluating their performances. This comes from the technical limitations due to the time taken for training each epoch on a multi-CNN VQ-VAE. This disconnects the runtime in Colab and prevents further training. However, the results are still comparable and thus the hypothesis that with full training, the multi-encoder can and will perform better than the single-encoder setup is not unfounded.

A lot of the viability issues of supervised learning techniques come from environment variability, Wi-Fi access point topology and physical interference. This holds in both supervised and unsupervised situations, but such datasets are extremely expensive to create and exhaustive data for such HAR just does not exist. Evaluation of such data would prove some hypotheses proposed both in this study and in the cited studies. A general idea can be accrued by running the model on the full OPERANet experiments because there are some variations in the experiment setups throughout, but viability can only be confirmed after a full release of such a trained system in a real-world situation for a study that includes access points shifts mid-experiment, physical obstructions introduced during the experiment and so on.

Hyperparameter tuning is an integral part of all model training. In the case of VQ-VAEs, however, the implications of the change to the codebook, learning rates and latent space dimensionalities are practically impossible to foresee other than some general idea. This is because of how the autoencoder works by principle, as well as the quantisation step, introducing some unpredictability in the way the latent space evolves from the beginning. Most of the hyperparameters used in this study were set according to previous works, but it would be worth doing some sweeps to figure out the best tuning for a specific codebook size. However, since VQ-VAEs are, by definition, a multi-objective chasing model, broader optimisation tweaks may shift the Pareto frontier and cause regressing performance (as seen in multi-modality without increasing codebook size).

Since the biggest draw of PWR is the idea of using existing Wi-Fi networks to perform HAR, one of the more variable objectives is to account for multiple devices and their bandwidths. Currently, the study is conducted on a single hardware stack, which is not the ideal experimental variability to be considered, especially considering the ideal outcome being system implementation in multiple hardware. So a more viable reconstruction of this study would use data from multiple hardware, normalised to form spectrograms that are then used to train a model that can construct a latent space that can perform HAR across multiple hardware stacks and bandwidths to a reasonable enough degree.

5.4 Future Work

As mentioned above, some more data gathering to account for real-world scenarios like multiple obstructions, multiple devices, and multiple sites would be worth prying into to address some generalisation concerns. The idea would be to conduct some contrastive supervised learning algorithms against the proposed VQ-VAE solutions, followed by some comparisons to concretise the claim of generalizability for unsupervised algorithms.

Since once formed, the latent encoders+quantizer step is fairly straightforward and fast, there could be a case made for the implementation of more complicated classifiers than a simple KNN or MLP, given the resolution of time within which HAR has to be conducted in a setting like assisted-living, which would likely be the most time-sensitive implementation.

A few-shot learning algorithm can also be tested on VQ-VAEs by means of adding a loss component that promotes class separation in the early stages of the training. This, combined with hyperparameter tuning, might give us a very well-clustered latent space, which could then be used via KNNs on averaged samples from each class. However, this would require extensive training and tuning, which are out of scope for the current implementation.

Bibliography

[1] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2911–2918, 2012.

[2] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv:1803.01271*, 2018.

[3] Mohammud J. Bocus, Wenda Li, Shelly Vishwakarma, Roget Kou, Chong Tang, Karl Woodbridge, Ian Craddock, Ryan McConville, Raul Santos-Rodriguez, Kevin Chetty, and Robert Piechocki. Operanet, a multimodal activity recognition dataset acquired from radio frequency and vision-based sensors. *Scientific Data*, 9, August 2022.

[4] Mohammud J. Bocus, Xiaoyang Wang, and Robert J. Piechocki. Streamlining multimodal data fusion in wireless communication and sensor networks. *IEEE Transactions on Cognitive Communications and Networking*, 10(1):252–264, 2024.

[5] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys*, 46(3):1–33, 2014.

[6] Victor C. Chen, F. Li, S.-S. Ho, and H. Wechsler. Micro-doppler effect in radar: Phenomenon, model, and simulation study. *IEEE Transactions on Aerospace and Electronic Systems*, 42(1):2–21, 2006.

[7] Grammarly. Autoencoders in deep learning: How they work and why they matter.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[9] J. A. Högbom. Aperture synthesis with a non-regular distribution of interferometer baselines. *Astronomy and Astrophysics Supplement Series*, 15:417–426, June 1974.

[10] Kiran Sankar E J. opportunisticwifiradarhar: Dissertation 2025. https://github.com/KIRANSANKAREJ/opportunisticWiFiRadarHAR, 2025. GitHub repository. Accessed 2025-08-22.

[11] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[12] Adrian Łańcucki, Jan Chorowski, Guillaume Sanchez, Ricard Marxer, Nanxin Chen, Hans J. G. A. Dolfing, Sameer Khurana, Tanel Alumäe, and Antoine Laurent. Robust training of vector quantized bottleneck models, 2020. Published at IJCNN 2020.

[13] Colin Lea, Michael D. Flynn, Rene Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks for action segmentation and detection. In *CVPR*, 2017.

[14] Wenda Li, Mohammud Junaid Bocus, Chong Tang, Shelly Vishwakarma, Robert J. Piechocki, Karl Woodbridge, and Kevin Chetty. A taxonomy of wifi sensing: CSI vs passive wifi radar. In *2020 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6, Taipei, Taiwan, December 2020. IEEE.

[15] Robert J. Piechocki, Xiaoyang Wang, and Mohammud J. Bocus. Multimodal sensor fusion in the latent representation space. *Scientific Reports*, 13:2005, 2023.

[16] Qifan Pu, Sidhant Gupta, Shyamnath Gollakota, and Shwetak Patel. Whole-home gesture recognition using wireless signals. In *ACM SIGCOMM*, pages 27–38, 2013.

[17] Farheen Ramzan, Muhammad Usman Ghani Khan, Asim Rehmat, Sajid Iqbal, Tanzila Saba, Amjad Rehman, and Zahid Mehmood. A deep learning approach for automated diagnosis and multi-class classification of alzheimer's disease stages using resting-state fmri and residual neural networks. *Journal of Medical Systems*, 44(2), 2020.

[18] Pawan Saxena. Variational autoencoders. Last updated: 15 Jul 2025.

[19] Sheng Tan, Yili Ren, Jie Yang, and Yingying Chen. A survey of commodity wifi sensing in 10 years: Current status, challenges, and opportunities. *IEEE Internet of Things Journal*, 2023.

[20] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.

[21] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.